Speciale 1

CHIP

Mensile di micro e personal computer

BASIC

corso di programmazione per microcalcolatori

BASIC

tecniche nuove

Supplemento a CHIP n. 2/84 periodico mensile

Mensile di micro e personal computer
ISSN 0392-9353

Direttore responsabile: Giuseppe Nardella - **Direttore editoriale:** Enzo Guaglione - **Redattore capo:** Richard Kerler - **Redazione:** Giorgio Panzeri - Maresa Colecchia - **Redazione libri:** Letizio Cacciabue - **Inchieste:** Carlo Gianola - **Coordimento tecnico:** Roberto Albanesi - Mario Monici - Lucia Maiandi - **Fotografia:** Ezio Geneletti - **Pubblicità:** Cesare Gnocchi - **Coordinamento annunci:** Carla Zoia - **Abbonamenti:** Lia Giaccherini, Daniela Conti - **Amministrazione:** Bianca Perazzini, Giorgio Montanari - **Convegni:** Luciana Vogogna - **CHIP è pubblicata in lingua italiana - tedesca - spagnola -** Tribunale di Milano n° 120 - 12.3.83 - Registro Nazionale della Stampa n° 630 v.7 f.233 - 16.12.82 - **Banche:** Banca d'America e d'Italia Agenzia I - Milano - **Distributore:** Messaggerie Periodici - ME.PE. Via G. Carcano, 32 - 20141 Milano - **Stampa:** Grafiche Jodice - Rosate - **Fotocomposizione:** Grafica Quadrifoglio S.r.l., P.zza Principessa Clotilde, 10 - 20121 Milano - Tel. 02/6599936 -

- **Editore:** TECNICHE NUOVE - Milano - **Direzione**, Redazione Amministrazione, Pubblicità, Abbonamenti: Editrice Tecniche Nuove, Via Moscova 46/9 - 20121 Milano - Tel. 02/6590351 (8 linee) - Telex: 334647 TECHS I - **Pubblicità e piccoli annunci:** - Milano - Via Moscova, 46/9 - Tel. 02/6590351 - Torino - C.so M. D'Azeglio, 60 - Tel. 011/682604-651674 - Vicenza - Corso Padova, 17/19 - Tel. 0444/512238 - Padova - Via Pellizzo, 14/1 - Tel. 049/772370 - Bologna - Via Imola, 13/A - Tel. 051/324311 - **Abbonamenti:** Italia: Lit. 30.000 annue - Estero Lit. 45.000 (Europa); Lit. 80.000 (Oltremare) - Fascicolo: Lit. 3.000 (arretrato Lit. 5.000). - Per abbonarsi a CHIP è sufficiente versare l'importo sul conto corrente postale n. 30754204 oppure a mezzo vaglia o assegno bancario intestati alla Casa Editrice Tecniche Nuove S.r.l. -Via Moscova, 46/9 - 20121 Milano. Gli abbonamenti decorrono dal mese successivo al ricevimento del pagamento. - Spedizione in abbonamento postale Gruppo III/70 - **U.S.P.I.** Associata all'Unione Stampa Periodica Italiana - **A.I.E.** Associazione Italiana Editori - **CHIP ha richiesto il controllo A.D.S.** - Accertamenti Diffusione Stampa.

PER PROGRAMMARE
PERFETTAMENTE
SENZA ESSERE
UN PROGRAMMATORE
PER APPROFONDIRE
IL DIALOGO COL
TUO ELABORATORE
PER PROGRAMMARE
L'ACQUISTO
DI UN PERSONAL
PER IMPARARE SUL SERIO
A FARE I VIDEO GIOCHI
PER ACQUISIRE
UNA PERFETTA
PADRONANZA
DEI LINGUAGGI
PER LEGGERE
I PROGRAMMI
TRA LE RIGHE
PER ELABORARE
LE TUE IDEE
MENSILE
DI MICRO E
PERSONAL
COMPUTER
GGK
TIENILO IN MEMORIA
CHIP E' UN PERIODICO TECNICHE NUOVE

Alain Checroun

BASIC

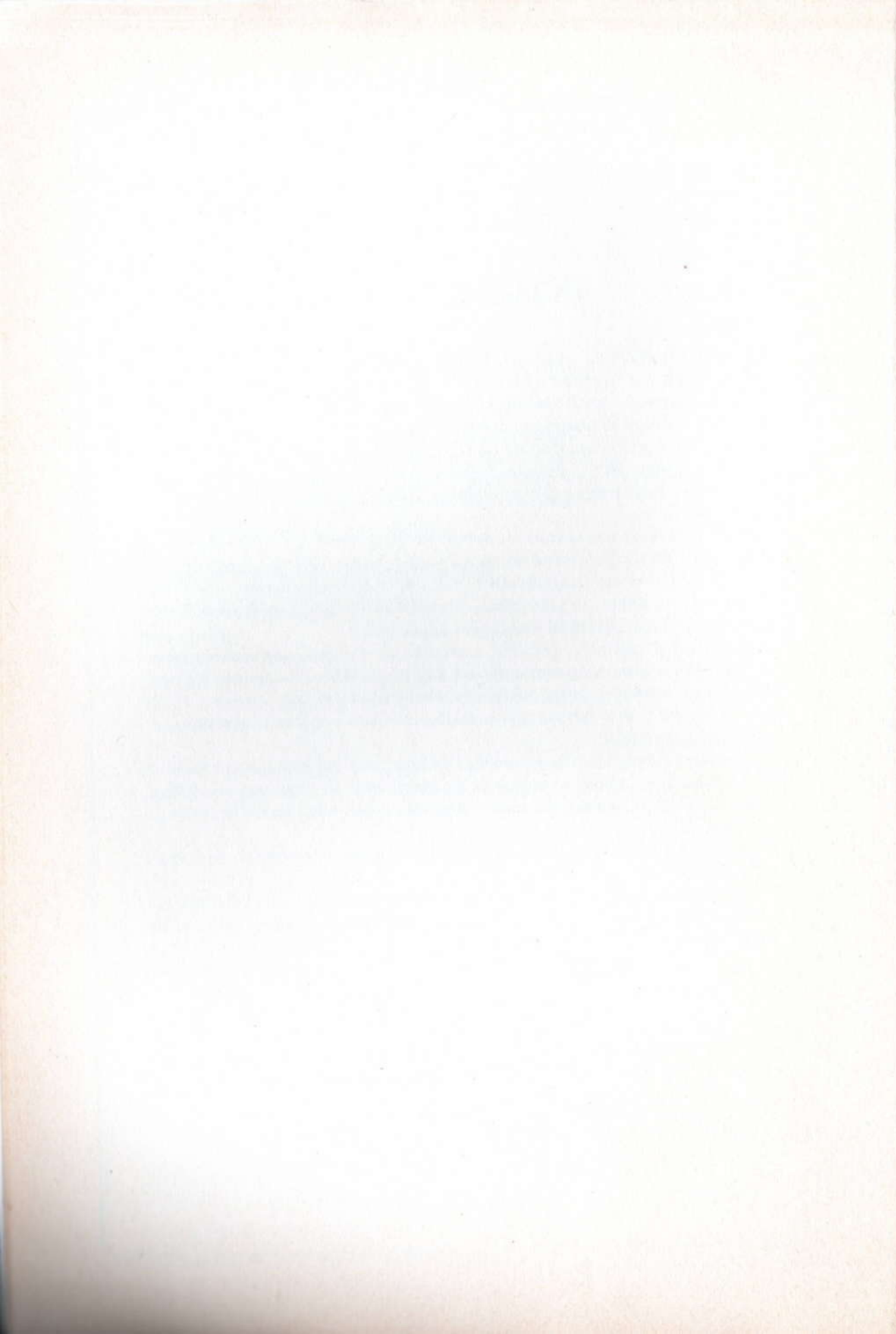
corso di programmazione dei microcalcolatori

tecniche nuove

INDICE

INTRODUZIONE

La microinformatica è una scienza applicabile alla nostra vita quotidiana sia professionale sia familiare. La dimensione ridotta, l'affidabilità, la facilità d'uso e il modesto costo mettono i piccoli calcolatori alla portata di tutti.
Abbiamo voluto spiegare il più chiaramente possibile questo fenomeno tecnico che provocherà senza dubbio un rinnovamento sociale importante a breve scadenza. Questo libro è dunque dedicato al lettore che non conosce o che conosce poco i calcolatori e che un giorno o l'altro si troverà nella condizione di doverli utilizzare.
Dopo un breve richiamo storico, definiamo gli elementi che costituiscono un sistema informatico costruito su un microcalcolatore, oltre ai rispettivi ruoli. Descriviamo in modo semplificato il principio di funzionamento e i principali concetti da ricordare, in particolare il concetto di linguaggio di programmazione e i differenti livelli di linguaggio applicabili.
Al momento attuale tutti i personal computer, oltre a molti calcolatori di grosse dimensioni, sono programmabili nel linguaggio Basic. Il capitolo 2 è dedicato alla spiegazione di questo linguaggio nella sua forma più corrente. I capitoli 3 e 4 sono utili al lettore che, avendo assimilato i concetti elementari, intendesse approfondirli.
Il capitolo 3 è dedicato alla descrizione di istruzioni più complesse, tipiche di alcune macchine. Abbiamo deciso di descrivere uno dei linguaggi più diffusi, il Basic Plus della Digital, ma molte istruzioni sono valide anche per altri costruttori.
Il capitolo 4 affronta il concetto di archivio e il modo di trattarlo con il Basic della Digital.
Tengo particolarmente a ringraziare Marc Podgrodski per la sua partecipazione alla stesura degli ultimi due capitoli, oltre che Pierre Durieux e Antoine Temime che hanno voluto rivedere e correggere taluni esercizi.

L'Autore

CAPITOLO 1

I MICROCALCOLATORI

1.1. DAI CALCOLATORI AI MICROCALCOLATORI

Come è noto, il calcolatore è costituito da elementi a base di componenti elettronici. Dagli anni cinquanta, in cui si fabbricavano calcolatori a tubi elettronici il cui elemento principale era l'Unità Centrale (aritmetica e logica) che richiedeva potenti sistemi di condizionamento e locali appositamente attrezzati fino ai nostri giorni, in cui l'unità di elaborazione è trascurabile in volume e in prezzo rispetto agli altri elementi del sistema, questi componenti hanno subito una progressiva miniaturizzazione.

Pur non entrando nel merito della tecnologia della miniaturizzazione, cercheremo semplicemente di dare un'idea del progresso compiuto citando i risultati ottenuti nei periodi che hanno segnato maggiormente l'evoluzione dei calcolatori. In primo luogo due precursori:

— 1944. Aiken ad Harvard completa il «MARK I»; combinando tubi e relais effettua un'operazione di somma in 0,25 s e una di moltiplicazione in 4 s.

— 1946. Eckert e Mauchly all'Università di Pennsylvania mettono a punto il primo calcolatore completamente elettronico: l'ENIAC (Electronic Numeric Integrator and Automatic Calculator). Pesa diverse decine di tonnellate, consuma l'energia elettrica di un treno e richiede un grosso impianto di condizionamento.
Duecento persone debbono sorvegliare il suo funzionamento... un guasto ogni dieci minuti... però esegue 5 mila somme al secondo.

— 1948. Dopo lo studio teorico d'importanza capitale compiuto da Von Neuman, esce l'EDVAC (Electronic Discrete Variable Automatic Calculator). La codifica binaria dell'informazione diventa preponderante.

— 1952. Descrizione dell'effetto di campo da parte di Shockley.

— 1956 ÷ 1968. Sostituzione dei tubi catodici con transistori. Introduzione di grosse memorie a ferrite. Linguaggi di programmazione. Da qui maggiore affidabilità, velocità, flessibilità d'uso. Dischi magnetici, grosse stampanti.

— 1965 ÷ 1970. Circuiti integrati, console di visualizzazione. Metodi conversazionali. Reti, elaborazione a distanza. Le prestazioni sono moltiplicate per 100, i prezzi divisi per 10.

— 1970 ÷ 1980. Progresso considerevole nella miniaturizzazione dei componenti e dei circuiti: il calcolatore in una scatola. Le prestazioni sono ancora moltiplicate e i prezzi divisi per 10.

Oggi si può trovare sul mercato un'unità centrale di 32 Kbyte munita di video, di tastiera, di due dischi flessibili da 180000 byte, di stampante, per una spesa attorno ai 5 milioni di lire e tale da occupare un volume equivalente a tre cassetti di scrivania.
È l'inizio dell'era dei microcalcolatori la cui unità centrale è realizzata con un circuito microprocessore in un'unica scatola contenente componenti LSI (Large Scale Integration). In una pasticca di silicio di qualche millimetro di lato si possono includere fino a 50000 transistori e le loro connessioni.

1.2. STRUTTURA DI UN MICROCALCOLATORE

Come tutti i calcolatori, i microcalcolatori debbono svolgere tre funzioni principali:

— una di commutazione (o elaborazione)
— una di memorizzazione
— una di scambio.

Queste tre funzioni sono realizzate da tre tipi di elementi che sono alla base della struttura del calcolatore:

— *l'unità centrale* (CPU o Central Processor Unit): ha la funzione di eseguire le elaborazioni indicate dalle istruzioni che essa ha trovato nella sua memoria (registri);
— *la memoria centrale*: serve a memorizzare i programmi (insieme di istruzioni) e i dati;
— *le apparecchiature di ingresso/uscita* (I/O o Input/Output): permettono la comunicazione con il mondo esterno. Ad esempio, una tastiera permette d'immettere i dati, una stampante di far uscire i risultati.

Riassumiamo queste definizioni fondamentali schematizzando i quattro elementi funzionali di ogni calcolatore:

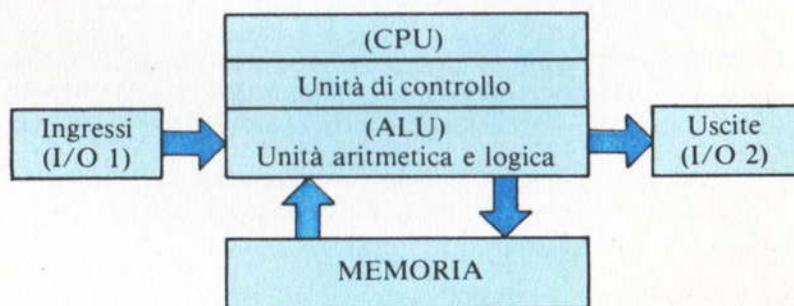



Fig. 1

Fisicamente un microcalcolatore è costituito da due o tre schede logiche e da un'alimentazione per la CPU (la memoria è compresa in questa). I terminali d'ingresso/uscita più diffusi sono costituiti da una tastiera e da un video. La tastiera (tipo macchina da scrivere) permetterà d'immettere dei caratteri alfanumerici e grafici (un carattere per tasto) codificati in binario; sul video saranno visualizzati i caratteri generati in uscita dal microcalcolatore.
Naturalmente potranno essere collegati dei terminali di tipo diverso in funzione dell'utilizzazione desiderata. Ad esempio, l'uscita su stampante sarà spesso utile per conservare traccia di certi risultati, stabilire la situazione, preparare la corrispondenza. In ingresso è facile intuire l'utilità degli organi che permettono la lettura rapida d'un gran numero d'informazioni: unità mini-disco (floppy) e bande magnetiche (minicassette).
I terminali previsti potranno essere collegati al microcalcolatore grazie ai canali (BUS). Questi sono le vie di comunicazione tra l'unità centrale e i differenti organi ad essa collegati. Un canale è un insieme di linee raggruppate per funzione.
Si trova generalmente:

— un canale dati
— un canale indirizzi
— un canale di controllo.

1.3. PRINCIPIO DI FUNZIONAMENTO

1.3.1. Rappresentazione delle informazioni

È bene ricordare che un microcalcolatore è una macchina destinata a manipolare informazioni di natura diversa.
Queste potranno essere:

— *dei dati* o elementi che definiscono il problema da risolvere. I gruppi di dati della stessa natura costituiscono gli archivi;
— *delle istruzioni* o indicazioni sulle elaborazioni da effettuare. La sequenza di istruzioni che conduce a una,certa elaborazione costituisce un programma.
In tutti i casi queste informazioni saranno ricondotte a una rappresentazione binaria all'interno della macchina. Questa limitazione è legata alla natura fisica dei supporti utilizzati. Al momento della progettazione il costruttore compie una scelta che definisce la struttura dell'informazione e che corrisponde ai circuiti cablati e/o microprogrammi previsti. È così che egli definirà l'insieme delle operazioni elementari possibili e il volume globale disponibile.
L'unità elementare d'informazione è il *bit* (contrazione di binary digit), ma ciascuna macchina sarà caratterizzata dalla più piccola quantità d'informazione accessibile direttamente (indirizzabile), ossia un insieme di bit definito *parola*. La maggior parte dei microcalcolatori sono macchine a parole di 8 bit, ma ne esistono anche con parole di 16 bit e di 32 bit.

Con parole di 8 bit si disporrà di un *vocabolario* potenziale di 2^8 = 256 parole differenti. Si comprende quindi l'interesse di disporre di macchine a parole di 32 bit, se non si è condizionati dal volume o dal costo.
Fissata la lunghezza delle parole, il costruttore definisce la struttura di un'istruzione. Si avrà ad esempio, nel migliore dei casi:

un codice operazione	un indirizzo operando 1	un indirizzo operando 2	un indirizzo risultato
8 bit	8 bit	8 bit	8 bit

quindi un'istruzione a tre indirizzi che richiede 32 bit. Poiché il codice operazione è di 8 bit, si disporrà al massimo di 256 operazioni differenti. Questo è più di quanto serve.
L'insieme dei codici operazione costituisce il vocabolario della macchina, ad esempio:

00010100	significherà sommare.

D'altra parte il costruttore definisce la capacità di memorizzazione della sua macchina prevedendo i supporti fisici corrispondenti. In generale, la memoria del calcolatore potrà contenere molte decine di migliaia di parole. Per questo motivo si è definita la macrounità di capacità che corrisponde a 1024 parole e che viene designata con K (per kilo). Si dirà, ad esempio, che si dispone di una memoria di 32 Kparole (condizione normale). I microcalcolatori correnti possono disporre di memorie di 64 o di 128 Kparole in versione commerciale.

1.3.2. Organizzazione della memoria

Se si schematizza la memoria con un rettangolo che contiene delle caselle numerate, si segneranno tante caselle quante sono le parole previste in questa memoria. Ad esempio, 32768 per una memoria di 32 Kparole.
Una parte di questa memoria conterrà dei programmi preregistrati dal costruttore, in particolare il programma capace d'interpretare le istruzioni dell'utilizzatore (programma di gestione). Un'altra parte conterrà il programma dell'utilizzatore.
Infine essa permette di memorizzare tutti i dati necessari.
Come s'è detto precedentemente, tutte le informazioni memorizzate sono ricondotte a *parole* (ad esempio di 8 bit, cioè dei byte) e a ciascuna di queste parole è assegnato un indirizzo; nel nostro esempio la prima casella avrà indirizzo 0, l'ultima indirizzo 255. Questi indirizzi si scriveranno in binario:

00000000 e 11111111

(numero massimo esprimibile con 8 bit).

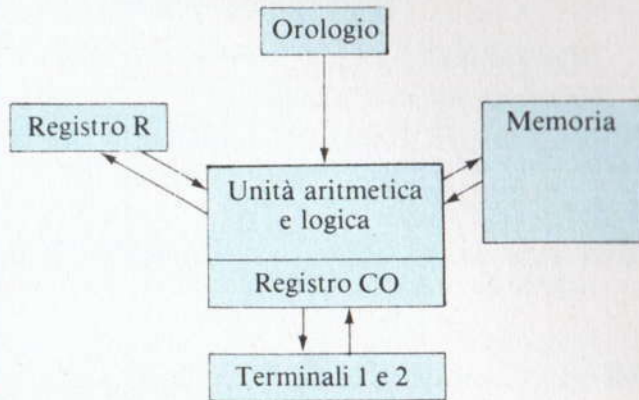



Fig. 2 **Fig. 3**

Nota. Le tecnologie utilizzate nei microcalcolatori sono diverse e l'utilizzatore è chiamato a confrontarsi con sigle che è opportuno impari sin dall'inizio. Sono utilizzati due tipi principali di memoria:

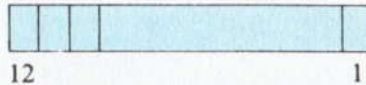
— le R.O.M. (Read Only Memory) o memorie a sola lettura destinate alla memorizzazione dei programmi che non saranno modificati. È il caso, ad esempio, del programma di controllo;
— le R.A.M. (Random Access Memory) o memorie ad accesso casuale.

Possono essere *scritte* o *lette*, ma presentano l'inconveniente d'essere volatili, cioè il loro contenuto si perde quando la tensione d'alimentazione viene interrotta.
Sarà quindi necessario che l'utilizzatore provveda a salvare le informazioni utili su un altro supporto prima di spegnere la macchina.

1.3.3. Esempio di funzionamento

Per comprendere il principio di funzionamento di un calcolatore descriveremo una configurazione semplificata, ma contenente gli elementi essenziali per trattare l'informazione. Essa è riassunta dallo schema di fig. 3.

— *L'orologio* (clock) è un contatore elettronico che invia impulsi regolarmente spaziati permettendo così di sincronizzare le azioni della CPU.

— *La memoria* è un insieme di 256 parole di 12 bit; a ciascuna parola è assegnato un indirizzo (da 0 a 255) (v. fig. 2).
— *Il registro* R è un gruppo di elementi elettronici che possono prendere due stati stabili, definiti 0 e 1. Vi sono in totale 12 bistabili in questo registro. Cioé esso permetterà di ricopiare integralmente una parola dalla memoria.

— *Il registro contatore* è un gruppo di 8 bit. Esso permette di rappresentare numeri binari che vanno fino a 11111111, cioé 255; esso conterrà l'indirizzo della parola da prendere in conto in memoria.

— *I terminali* 1 *e* 2 sono, ad esempio, una tastiera che consente d'immettere le informazioni e una stampante che consente di stampare i risultati.
— *L'unità di calcolo* (CPU), sotto l'azione dell'orologio e seguendo un programma d'azione preregistrato o cablato, può accedere:
— allo stato d'una parola di memoria
— allo stato di R
— allo stato del registro contatore
— a tutte le informazioni provenienti dai terminali d'ingresso

Essa può anche modificare il contenuto di R e del contatore, oltre che comandare la stampa di caratteri sul terminale d'uscita.

Ipotesi di funzionamento

Con la messa in tensione viene lanciato automaticamente un certo numero d'azioni elementari:

Azione 1: il contenuto del registro contatore assume il valore 32
Azione 2: lettura del contenuto del registro contatore
Azione 3: lettura della parola il cui indirizzo è dato dal registro contatore
Azione 4: realizzazione dell'elaborazione che dipende dal contenuto della parola di memoria appena letta
Azione 5: il contenuto del contatore è aumentato di uno; si ritorna al primo passo

D'altra parte il costruttore ha anche definito l'insieme delle istruzioni possibili (cablate o microprogrammate) riassunte nella tabella seguente:

CONTENUTO DELLA CASELLA PUNTATA DAL CONTATORE	AZIONE RISULTANTE
1010 aaaaaaaa	Ricopiare nel registro R il contenuto della parola d'indirizzo A. Notazione simbolica (A) → R
1011 aaaaaaaa	Ricopiare nella parola d'indirizzo A il contenuto di R. (R) → A
1000 aaaaaaaa	Sottrarre il contenuto di R dal contenuto della parola d'indirizzo A e trascrivere il risultato in A. (A) — (R) → A
1101 aaaaaaaa	Somma del contenuto di R al contenuto della parola d'indirizzo A e trascrizione del risultato nella stessa parola. (R) + (A) → A
1100 aaaaaaaa	Sottrazione di 1 dal contenuto della parola d'indirizzo A e copia del risultato nella stessa parola. (A) — 1 → A
0001 aaaaaaaa	Iscrizione della quantità A nel registro contatore. A → CO
0101 aaaaaaaa	Iscrizione della quantità A nel registro contatore nel caso in cui la parola all'indirizzo 1 contenga 0, altrimenti nessuna azione. sì: A → CO (1) = 0 no: nessuna azione
011100000001	Uscita del numero contenuto nel registro R sul terminale 1.
011000000010	Attesa della battuta di un tasto sul terminale 2 e, quando questo avviene, iscrizione in R del codice binario corrispondente a questo tasto.
000000000000	Arresto della macchina (interruzione delle comunicazioni tra orologio e unità di calcolo).

Fig. 4

Si noterà che ciascuna istruzione è costituita da un codice operazione di 4 bit e da un indirizzo di 8 bit.
Per sintetizzare la spiegazione, si è seguita la convenzione seguente: *n* rappresenta l'indirizzo ed (*n*) il contenuto della parola di indirizzo *n*.
La freccia → significa «ricopiare in».
Supponiamo inoltre di aver registrato in memoria la sequenza di istruzioni seguente:

Indirizzo	Contenuto della parola
16	0000 0000 0000
32	0110 0000 0010
33	1011 0000 0001
34	1010 0001 1100
35	1011 0000 0010
36	0101 0010 1000
37	0110 0000 0010
38	1101 0000 0010
39	1100 0000 0001
40	0001 0010 0011
41	1010 0000 0010
42	0111 0000 0001
43	0000 0000 0000

Fig. 5

N. caselle indicate da	Azioni	Effetto sul contenuto delle caselle n. 16, 1 e 2 e dei registri R e R.C.				
		R.C.	16	1	2	R
32	Attesa del 1° carattere (qui 2)					
	2 → R	32	0	?	?	2
33	(R) → 1	33	0	2	?	2
34	(16) → R	34	0	2	?	0
35	(R) → 2	35	0	2	0	0
36	Test di (1) : (1) ≠ 0					
	Nessuna azione	36	0	2	0	0
37	Attesa di un carattere (qui 4)					
	4 → R	37	0	2	0	4
38	(R) + (2) → 2	38	0	2	4	4
39	(1) — 1 → 1	39	0	1	4	4
40	35 → CO	35	0	1	4	4
36	Test di (1) : (1) ≠ 0					
	Nessuna azione	36	0	1	4	4
37	Attesa di un carattere (qui 5)					
	5 → R	37	0	1	4	5
38	(R) + (2) → 2	38	0	1	9	5
39	(1) — 1 → 1	39	0	0	9	5
40	35 → CO	35	0	0	9	5
36	Test di (1) : (1) = 0					
	da cui					
	40 → CO	40	0	0	9	5
41	(2) → R	41	0	0	9	9
42	Stampa di R: la telescrivente					
	stampa il carattere 9	42	0	0	9	9
43	Fine delle operazioni					

Fig. 6

Se si battono sulla tastiera i numeri successivi:

2, 4 e 5

si vedrà apparire sulla stampante: 9.

Analisi del funzionamento (si veda la figura 6)

Con la disposizione della figura 5 in memoria, se l'utente avesse battuto 3-2-1-4 il calcolatore avrebbe stampato 7. In modo generale il calcolatore calcolerà la somma di un numero qualunque di cifre. La prima cifra *n* che l'operatore batte sul terminale rappresenta il numero di queste cifre, e le cifre seguenti sono quelle sulle quali viene calcolata la somma. La macchina stampa il risultato quando sono state sommate *n* cifre. *La disposizione binaria della figura 5 costituisce il programma della macchina. Scriverlo sul calcolatore significa programmarlo.*
In effetti la sequenza delle azioni precedenti si traduce in:

— registrazione nella casella 1 (contatore) del numero *n*
— azzeramento della casella 2 (totalizzatore)
— fintanto che il contatore non contiene zero, lettura della cifra seguente, aggiunta di questa cifra al totalizzatore (operazione della casella 38), sottrazione di uno al contenuto del contatore
— stampa del contenuto del totalizzatore (casella 2), quando il contatore (casella 1) contiene zero (le *n* cifre sono state sommate).

1.4. I DIFFERENTI LIVELLI DI LINGUAGGIO

1.4.1. Il linguaggio macchina

Come abbiamo avuto modo di dire a proposito delle descrizioni precedenti, tutti i calcolatori posseggono un vocabolario elementare definito dal suo costruttore e riconducibile a qualche decina di parole binarie corrispondenti all'insieme delle azioni che la macchina è in grado di compiere.
La combinazione di queste parole secondo una data regola di scrittura, ugualmente definita dal costruttore, costituisce un programma. Ad esempio, la figura 5 è un programma in linguaggio macchina (caratteristico della macchina semplificata descritta precedentemente). Ciascuna macchina ha il suo linguaggio, e un programma scritto per l'una non potrà essere utilizzato per un'altra.

1.4.2. I linguaggi d'assemblaggio

È evidente che la manipolazione da parte dell'uomo di un tale linguaggio non

è cosa molto pratica. Per rendere il linguaggio meno ostico e più mnemonico si è in primo luogo immaginato di designare:

— ciascun codice operazione
— ciascuna casella di lavoro
— ciascuna variabile

con gruppi di caratteri costituenti un codice mnemonico. Si avrà ad esempio:

Codice binario	Codice mnemonico	Significato
1010	LEGG	(A) → R
1011	SCRI	(R) → A
1000	SOTT	(A) — (R) → A
1101	ADDI	(R) + (A) → A
1100	DECR	(A) — 1 → A
0001	SALT	salto incondizionato
0101	SAZE	salto se zero
0111	USCI	uscita
0110	INGR	ingresso
0000	FINE	arresto

Fig. 7

Il programma della figura 5 diventerebbe, ad esempio, quello di fig. 8.

Questa presentazione è sotto forma di archivio. È esattamente ciò che s'immetterà in macchina e che verrà tradotto dall'*assemblatore*. Questo *assemblatore* è un programma già caricato in memoria e che tradurrà questo archivio in linguaggio binario (*in questo caso otterrà il programma descritto nella figura 5*).
L'assemblatore analizzerà la sequenza di caratteri dell'archivio di figura 8 nel modo seguente.

— VARIA: assegnazione delle caselle 1,2, ... ai gruppi di caratteri specificati nel paragrafo VARIA.
— CONST: assegnazione delle caselle 16, 17 ai gruppi di caratteri specificati nel paragrafo COST e scrittura dei valori di queste costanti (indicate in terza colonna) nelle caselle corrispondenti.
— OPER: scrittura del programma propriamente detto in binario a partire dalla casella 32.

Questo implica *per ciascuna linea* (ciascuna istruzione):

• Se vi sono caratteri sulla *prima colonna* (caratteri diversi dalla spaziatura), assegnare a questo gruppo di caratteri l'indirizzo della casella di memoria dove l'assemblatore scriverà il codice binario dell'operazione corrispondente alla linea.

1ª Colonna	2ª Colonna	3ª Colonna	
	PROGR	SOMMA	Dichiarative
	VARIA		
CØMPT			Dati
TØTAL			
	CØNST		
ZERO		0	Costanti
	OPER		
	ENTR	2	
	ECRE	CØMPT	
	LIRE	ZERØ	
	ECRE	TØTAL	
BOUCLE	SOZE	FINALG	
	ENTR	2	
	ADDM	TØTAL	Operazioni
	DECR	COMPT	
	SAUT	BOUCLE	
FINALG	LIRE	TØTAL	
	SØRT	1	
	FINI		
	FINA		

Fig. 8

• Tradurre i caratteri della *seconda colonna* nel codice binario dell'operazione corrispondente e scrivere questo codice all'inizio della parola di memoria (bit 12-9).

• Tradurre i caratteri della *terza colonna:*

- se sono cifre, tradurre il numero in codice binario e trascriverlo nei bit 8-1.

- Se questo gruppo di caratteri corrisponde al nome di una variabile, di una costante o di una casella d'operazione (in tutti i casi ciò che è scritto in prima

colonna), tradurre l'indirizzo di registrazione corrispondente al codice binario e scriverlo nei bit 8-1.

— FINA: interrompere l'assemblaggio, l'archivio è terminato.

Nell'esempio precedente abbiamo presentato un linguaggio Assembler e le azioni del programma d'assemblaggio corrispondenti. Si può già constatare un miglioramento nella comprensione del programma da parte dell'utilizzatore del calcolatore. I codici operazione scritti in forma mnemonica (LEGG, INGR, ecc) e i nomi delle caselle di memoria (variabili, costanti o indirizzi di programma) sono molto più espliciti.

1.4.3. Linguaggio avanzato

Si suole rendere il programma ancora più facile da manipolare scrivendolo in un linguaggio che ricordi il più possibile la lingua naturale (inglese, francese, italiano...).
Il linguaggio Basic è un esempio di tale linguaggio: se si vuole tradurre in Basic il programma che è servito d'esempio fino ad ora si ottiene:

```
10 INPUT C
20 LET T = 0
30 FOR I = 1 TO C
40 INPUT N
50 LET T = T + N
60 NEXT I
70 PRINT T
80 STOP
90 END
```

cioé:

10 Immissione di C (contatore)
20 Azzeramento di T (totale)
30 I = 1 prima dell'ingresso nel ciclo iterativo, se I > C si esce dalla iterazione e si va al numero 70.
40 Immissione della prima cifra N
50 Somma a T
60 Incremento di I e salto all'inizio della iterazione se I < C
70 Uscita di T (totale)
80 Codice di fine
90 Codice di fine archivio

Per ottenere tale programma in linguaggio macchina è necessario tradurlo per mezzo di un programma speciale definito *compilatore*; questo analizza ciascuna istruzione, segnala gli eventuali errori di sintassi con dei messaggi di errore e scrive in memoria il corrispondente programma binario.

CAPITOLO 2

IL BASIC

Al fine di rendere più efficace la presentazione di questo capitolo, abbiamo adottato un approccio a spirale, cioè partiamo da esempi elementari e procediamo verso un ampliamento del linguaggio ricorrendo a esempi di complessità crescente. Questo obbliga il lettore a rivedere più volte gli stessi concetti.

2.1. PRIMI CONCETTI DEL LINGUAGGIO

Istruzioni, Programma, Variabile, Dato

Battere sulla tastiera

```
PRINT 23-9
```

Premere il tasto RETURN
Si ottiene sullo schermo (o sulla stampante)

```
14
READY
```

Si era scritta un'*istruzione* (o *ordine BASIC*) che è stata interpretata ed eseguita dal compilatore immediatamente dopo che il tasto RETURN è stato premuto. Si tratta di un'istruzione detta *ordine diretto*. Ciò illustra l'importante vantaggio offerto dal Basic come linguaggio conversazionale: ogni volta che viene premuto il tasto RETURN, l'insieme dei caratteri battuti sulla tastiera è considerato un'istruzione e interpretata. Se questa interpretazione è positiva (non vi sono errori) e se nessun'altra informazione risulta necessaria, l'istruzione viene eseguita. Sarà così ogni volta che si vorrà eseguire una sola istruzione. Però i problemi più semplici richiedono molte istruzioni e i più complessi possono arrivare a migliaia di istruzioni. L'insieme delle istruzioni necessarie alla risoluzione di un dato problema costituirà un *programma*. Esaminiamo un esempio semplice.

Problema 1

Calcolare il quadrato e il cubo dei primi numeri interi:

Programma 1:

```
10 READ N
20 LET C = N * N
30 LET K = C * N
40 PRINT N,C,K
50 GOTO 10
60 DATA 1,2,3,...,9
70 END
RUN
```

Commenti:

— questo programma consiste in sette linee numerate di 10 in 10. Ciascuna linea rappresenta un'istruzione e i numeri di linea servono a precisare l'ordine in cui le istruzioni debbono essere eseguite (si può arrivare fino a 99999). Così si sarebbe potuto scrivere senza variare il programma:

```
10 READ N
60 DATA 1,2,3,...,9
40 PRINT N,C,K
20 LET C = N * N
```

Evidentemente si possono numerare le linee in ordine progressivo, però la numerazione di 10 in 10 permette una maggiore flessibilità e quindi di inserire in ogni momento istruzioni dimenticate o modifiche dei programmi. Ad esempio:

```
65 DATA 10,11,12,...,19
```

permette di aggiungere una sequenza di numeri al di là di quanto previsto nel programma originale.
— ciascuna istruzione comincia con un *ordine* che è rappresentato da una parola inglese che indica il tipo di azione che dev'essere eseguita.
— i caratteri che seguono questo *ordine Basic* sono simboli che rappresentano dei *dati*, delle *variabili* o degli *operatori*.
Così nell'istruzione:

```
30 LET K = C * N
```

30 è il numero dell'istruzione
LET è l'ordine Basic
K,C e N sono delle variabili
= e * sono degli operatori aritmetici
— gli spazi tra i differenti caratteri d'una linea non sono presi in considerazione dal compilatore e servono solo a facilitare la lettura del programma. Si può scrivere ad esempio:

```
10READN
20LETC=N*N
```

Tra i 5 tipi di istruzioni che figurano nell'esempio precedente, alcuni meritano qualche commento particolare.
L'*ordine* READ (istruzione 10) è un ordine di lettura e dev'essere sempre accompagnato da istruzioni DATA. Queste sono in effetti degli archivi di dati creati al momento della compilazione e consultati sequenzialmente, cioé il *calcolatore* quando incontra l'ordine READ preleverà il primo valore dall'archivio e lo assegnerà alla variabile N. Nel seguito del programma si avrà dunque N = 1 fintanto che non vi sia un nuovo ordine READ o un'istruzione LET N = (si veda oltre).
L'*ordine* LET (istruzioni 20 e 30) permette d'effettuare operazioni aritmetiche e d'inviare il risultato nella casella di memoria il cui simbolo è rappresentato dal simbolo situato a sinistra di =. Ad esempio, l'istruzione 20 effettua il prodotto (*) di N per N ed assegna il risultato alla variabile C. Evidentemente non vi potrà essere che una sola variabile alla sinistra del segno uguale.
Vediamo apparire qui il concetto di «variabile», cioé l'utilizzazione di simboli che sono interpretati dal compilatore come indirizzi di memoria.
È sul contenuto di questi indirizzi che si applicano tutte le operazioni previste nel programma. In Basic una variabile è rappresentata da una sola lettera seguita da una sola cifra, ad esempio: N,X,O2 (lettera O), Y5.
Gli operatori aritmetici usuali sono rappresentati in Basic dai simboli seguenti:

	Parentesi	( )
	Elevazione alla potenza	↑
(stesso peso)	Quoziente	/
	Prodotto	*
(stesso peso)	Differenza	—
	Somma	+

Li abbiamo elencati in ordine di peso decrescente. Cioé in una espressione aritmetica sono le parentesi più interne ad essere calcolate per prime.
All'interno delle parentesi, si effettuano le potenze, poi i quozienti e i prodotti, quindi le somme e le differenze. Nel caso in cui 2 operatori, in una data parentesi, hanno lo stesso peso, si comincia da quello situato più a sinistra.
Esempio: se si deve esprimere in Basic:

$$h = \frac{\dfrac{a + b}{c - d}}{e \cdot f} (x + y)^3$$

si ha:

```
20 LET H = ( (A + B)/( (C — D)/(E — F) ) ) * (X + Y) ↑ 3
```

Può meravigliare il numero di parentesi utilizzate in Basic in confronto a quello dell'espressione data. Questo dipende dal fatto che nel primo caso utilizziamo una rappresentazione a due dimensioni (la posizione dei simboli nel piano li informa dell'ordine in cui si debbono applicare gli operatori), mentre la scrittura Basic deve rispettare la natura unidimensionale dell'informazione che dev'essere presentata al compilatore. Abbiamo aggiunto delle frecce per precisare come le parentesi debbano essere appaiate. In effetti non si crea confusione se si decide di accoppiare la prima parentesi chiusa con la prima parentesi aperta che la precede. Una volta eseguito il contenuto della parentesi, si cancella la parentesi e si passa alla seguente per la quale si procede nello stesso modo. Il compilatore utilizza un algoritmo di questo genere per la traduzione dell'espressione, cioé al momento della scomposizione dell'istruzione Basic in istruzione elementare binaria.

Note

— È spesso possibile utilizzare meno parentesi grazie alla convenzione della priorità delle operazioni. Nell'esempio precedente si poteva scrivere:

```
20 LET H = (A + B) * E * F/(C — D) * (X + Y) ↑ 3
```

— Si può anche risparmiare sul numero delle operazioni da effettuare in una espressione, modificando semplicemente l'ordine in cui si scrivono le operazioni (quando la permutazione è possibile).

— È bene verificare ogni volta che vi sia lo stesso numero di parentesi aperte e chiuse.

— Insistiamo sul fatto che il valore del risultato dell'espressione aritmetica situata a destra del segno = è assegnata alla variabile che si trova alla sinistra. Così un'espressione del tipo:

```
50 LET I = I + N
```

è perfettamente ammessa. Essa realizza l'incremento della variabile I di N unità. Se I avesse il valore 30 e N il valore 6, dopo l'esecuzione dell'istruzione 50 I avrebbe il valore 36.

— La maggior parte dei compilatori Basic permette di utilizzare indifferentemente le istruzioni aritmetiche con l'ordine LET o senza quest'ordine. Nel seguito scriveremo semplicemente, ad esempio:

```
50 I = I + N
```

L'ordine LET è sottinteso.

L'ordine PRINT (istruzione 40) non avrebbe bisogno di commenti: permette di stampare i valori assunti dai simboli N,C e K all'istante considerato. Vedremo nel seguito come è possibile prevedere l'impaginazione dei risultati.

L'ordine GOTO (istruzione 50) rinvia alla linea 10, cioé alla lettura del dato seguente: N che prenderà successivamente il valore 2, 3, 4,...50; questa istruzione permette la ripetizione del calcolo precedente con valori differenti. È il salto incondizionale; permette di effettuare delle iterazioni, cioé delle ripetizioni di una stessa parte del programma. Vedremo più avanti altri metodi per realizzare queste iterazioni di programma.

— Il calcolatore si ferma quando non trova ulteriori dati nell'archivio DATA (vedremo più avanti che è più sicuro controllare l'arresto del programma piuttosto che aspettare lo svuotamento dell'archivio dati).

— Abbiamo visto che ogni ordine READ comanda la lettura di un dato dall'archivio DATA nell'ordine in cui sono registrati i valori. È dunque importante verificare la corrispondenza tra gli ordini di lettura e la successione dei dati in DATA. Inoltre, invece di scrivere:

```
60 DATA 1,2,3,4,...,50
```

si sarebbe potuto scrivere anche:

```
60 DATA 1,2,3,4,5
61 DATA 6,7,8,9,10
.......
69 DATA 46,47,48,49,50
```

L'ordine END indica, per il compilatore, che si è raggiunta l'ultima istruzione del programma: di conseguenza ogni programma Basic dovrà contenere END come ultima istruzione.
Il premere RETURN dopo l'ultima istruzione del programma 70 END non ha alcun effetto. Per ottenere l'esecuzione è necessario battere il comando RUN (si tratta di un comando di sistema che non richiede un numero di linea). Così, contrariamente agli ordini diretti, le istruzioni d'un programma sono interpretate al momento dell'immissione (per mezzo di RETURN), ma non vengono eseguite che dopo il comando RUN.

1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	452
9	81	729

?OUT OF DATA ERROR IN 10
READY

L'ordine GOTO *n* è un ordine di salto incondizionale. Quando si arriva a questa istruzione, il seguito del programma riprende alla istruzione *n* che può essere a monte o a valle di questa.
Nel nostro esempio l'istruzione GOTO 10 rinvia alla linea 10, cioé alla lettura dei tre dati seguenti nell'archivio DATA. Quando questo è stato letto interamente, il programma reclama dunque ancora dei dati. E questo spiega il messaggio di errore che compare dopo il risultato.

2.2. IMMISSIONE DEI DATI E PRESENTAZIONE DEI RISULTATI

Utilizzazione degli ordini INPUT, READ, PRINT, DATA.

Problema 2

Si vuole creare un archivio in cui si registrano gli ordini di N clienti. Le informazioni per ciascun cliente sono le seguenti:

— *numero dell'ordine C (numero intero di tre cifre)*
— *quantità ordinata X (valore che può arrivare fino a 10000)*
— *prezzo unitario Y (5 cifre)*

Una volta creato l'archivio si desidera mostrare il suo contenuto sul video; ciascuna linea rappresenta un ordine con i valori X ed Y corrispondenti.

Programma 2:

```
 10 INPUT N
 20 DATA 1, 2450, 3030, 2, 5600, 1150, 3, 256, 724, 4,
 ..........
 ..........
 50 DATA 100, 2314, 1810
 60 READ C,X,Y
 90 PRINT C,X,Y
 95 GOTO 60
110 END
```

Commenti

— L'ordine INPUT è un'istruzione d'immissione dati in modo conversazionale. Quando il programma incontra quest'ordine s'interrompe per domandare all'utilizzatore d'immettere il valore corrispondente alla variabile N. Un punto interrogativo appare sullo schermo; quando viene immesso il valore, l'esecuzione del programma riprende. È possibile immettere i valori di più variabili per mezzo d'una istruzione INPUT, separandoli mediante una virgola:

```
10 INPUT A,B,C
```

Se al momento dell'esecuzione, quando compare il punto d'interrogazione, si batte 10, 30, 80 l'effetto è di assegnare:

il valore 10 alla variabile A
il valore 30 alla variabile B
il valore 80 alla variabile C

Nel nostro esempio, se si vogliono considerare 100 ordini, alla comparsa del punto d'interrogazione si batterà 100.
— La costituzione d'un archivio si riconduce all'immissione dei dati successivi corrispondenti ai numeri d'ordine, alle quantità da ordinare, al prezzo unitario.
DATA è l'ordine che permette di creare un tale archivio.
Bisogna notare che ciascun valore successivo all'ordine DATA è *separato dal seguente da una virgola*; nessuna virgola dopo l'ultimo numero d'una linea, salvo nel caso in cui il numero di valori da immettere superi la capacità della linea. In questo caso si continua sulle linee seguenti.
Esempio

```
100 DATA 25, 4870, 1385, 25684300, 290125
144, 294, 3249, 57856
100 READ, A,B,C,D,E,F,G,H
```

avrà come conseguenza di mettere

```
       25 in A
     4870 in B
     1385 in C
 25684300 in D
290125144 in E
      294 in F
     3249 in G
    57856 in H
```

Un'istruzione DATA non è eseguibile. Essa dovrà sempre essere accompagnata da un'istruzione READ.
— *L'ordine* READ determina la lettura dell'archivio DATA secondo l'ordine sequenziale delle informazioni che vi figurano. Nel nostro esempio i 3 primi valori incontrati saranno assegnati alle variabili C, X, Y. Si avrà dunque in memoria:

C = 1   X = 2450   Y = 3020

L'istruzione GOTO 60 va a produrre una nuova lettura dei 3 numeri consecutivi, fino all'esaurimento dell'archivio.

Anche in questo caso è necessario separare le variabili con virgole.
L'istruzione PRINT C,X,Y lancerà la stampa sulla stessa linea dei valori istantanei delle variabili C,X,Y, cioé:

```
1          2450          3020
```

Si possono scrivere in questo modo fino a 4 risultati su una stessa linea (limite dello schermo che è diviso in 4 zone dalla utilizzazione di virgole tra le variabili). Si può arrivare fino ad 8 risultati per linea utilizzando dei «;» al posto di «,».
L'istruzione 95 GOTO 60 determina, come si è visto per READ, la stampa ripetuta di 3 numeri consecutivi.
In definitiva si vedrà apparire sullo schermo la tabella:

```
  1     2450     3020
  2     5600     1150
  3      256      724
  4       ,        ,
  ,       ,        ,
  ,       ,        ,
100     2314     1810
? OUT OF DATA ERROR IN 60
```

Ritroveremo il messaggio d'errore che segnala che l'archivio è stato esaurito e che il programma reclama altri dati.

2.3. ISTRUZIONI DI TEST O SALTO CONDIZIONATO

Problema 2bis

A partire dall'archivio definito nel problema 2, calcolare la somma S = X + Y e stampare le informazioni che riguardano i soli ordini tali che S sia inferiore a 3000.

Programma 2bis

```
10 INPUT N
20 DATA .....
............
50 DATA
60 READ C,X,Y
90 PRINT, C,X,Y
55 PRINT «PICCOLI ORDINI»
57 I = 1
65 I = I + 1
70 S = X + Y
80 IF S > = 3000 THEN 95
95 IF I < N THEN 60
110 END
```

Commenti

Per rispondere alla domanda posta abbiamo aggiunto le istruzioni 55,57,65,70,80 e abbiamo cambiato l'istruzione 95.
Quest'ultima, 95 GOTO 60, era un'istruzione di salto incondizionato: abbiamo introdotto invece un'istruzione di *salto condizionato*. Al momento dell'esecuzione dell'istruzione:

```
80 IF S > = 3000 THEN 95
```

la macchina confronta il valore calcolato di S con il numero 3000. Se il valore di S è superiore o uguale a 3000 l'istruzione da eseguire sarà la 95, altrimenti (se S è minore di 3000) la macchina eseguirà l'istruzione che segue:

```
90 PRINT C,X,Y
```

Notiamo a questo proposito che l'ordine sequenziale corrisponde, come è stato detto in precedenza, all'ordine crescente dei numeri di istruzione. Così, ogni volta che S sarà inferiore a 3000, si stamperanno su una linea i valori di C,X,Y.
All'istruzione 95 si effettua un nuovo confronto per interrompere l'elaborazione al termine di un numero determinato di iterazioni.
È per contare queste iterazioni che si sono utilizzate le istruzioni:

```
57 I = 1                    (utilizzazione di un contatore)
65 I = I + 1                (incremento di questo contatore)
95 IF I < N THEN 60         (controllo di arresto)
```

In effetti quando I = N si passa all'istruzione in sequenza, cioè:

```
110 END
```

La forma generale di un'istruzione IF è:

N. linea	IF	Condizione da verificare	THEN	n. di linea ove passare se la condizione è verificata

↑ La condizione da verificare richiede 3 elementi: un «soggetto», una «relazione», un «oggetto».

Esistono 6 relazioni possibili:

= uguale
> maggiore
< minore
> = maggiore o uguale

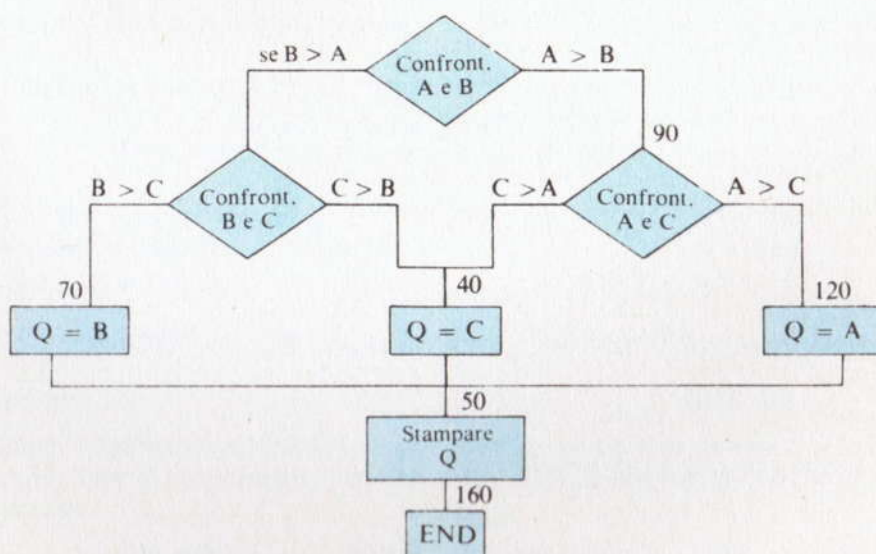
< = minore o uguale
< > differente

Il soggetto e l'oggetto possono essere variabili, espressioni o numeri. Ad esempio:

```
A5 > ((B + C)/K) * (M + P)
S * (Q + P) ↑ 2 = 15.2
124 < Y
MAR < > MAX
```

Possono essere presenti in un programma numerose istruzioni IF.

Problema 3

Si vuole determinare qual è il più grande tra i numeri positivi A, B, C seguendo il diagramma rappresentato qui sotto:

Programma 3

```
10 READ A,B,C
20 IF A > B THEN 90
30 IF B > C THEN 70
40 Q = C
50 PRINT Q
60 GOTO 160
70 Q = B
80 GOTO 50
90 IF A > C THEN 120
100 GOTO 40
120 Q = A
130 GOTO 50
150 DATA 3,1,4
160 END
```

Ritorno sull'ordine PRINT

Si è potuto vedere che il programma 2bis contiene l'istruzione 55 PRINT «PICCOLI ORDINI».
Esso permette di visualizzare sullo schermo dei commenti ai risultati e, nel caso generale, qualunque testo alfabetico che si desideri (è sufficiente inscriverlo tra apici).
Si potrà anche ottenere la tabella seguente:

—LISTA DEGLI ORDINI—

N° ord.	X	Y
1	2450	3020
2	5600	1150
,	,	,
,	,	,

utilizzando le istruzioni:

```
55 PRINT «LISTA DEGLI ORDINI»
56 PRINT «....N°....ord....X.....Y....»
```

Il solo ordine PRINT determina il salto d'una linea. Si potranno miscelare i testi e le variabili con un solo ordine PRINT.
Ad esempio:

```
70 PRINT «C = », C, «X = », X, «Y = », Y
```

2.4. LE ITERAZIONI DI PROGRAMMA

Il problema 2bis poteva ammettere il programma Basic seguente:

Programma 2ter

```
 20 DATA....
 .........
 55 PRINT «PICCOLI ORDINI»
 57 FOR I = 1 TO N
 60 READ C,X,Y
 70 S = X + Y
 80 IF S > = 3000 THEN 95
 90 PRINT C,X,Y,S
 95 NEXT I
110 END
```

L'istruzione 57 FOR I = 1 TO N significa: «eseguite tutte le istruzioni che seguono fino all'istruzione NEXT, e questo N volte».
Essa realizza l'iterazione di programma definita al paragrafo precedente; il contatore viene creato, incrementato e confrontato con il limite in modo implicito.
La forma generale dell'istruzione è:

```
20 FOR I = I1 TO I2 STEP S
    ''
    ''
    ''
130 NEXT I
```

che si tradurrà letteralmente con:
«Con l'indice I che va da I1 a I2 con passo S, eseguire le istruzioni che seguono fino ad incontrare l'istruzione NEXT».
Notiamo che l'indice utilizzato nell'istruzione FOR dev'essere ripetuto in NEXT. Gli indici debbono essere degli interi o dei reali positivi, negativi o nulli; possono anche essere sostituiti da una espressione aritmetica o da una variabile.

Esempi

```
25 FOR K = 10 TO 15
------------------------
34 FOR N5 = 50 TO 1 STEP -1
------------------------
48 FOR J = 3,5 TO 7,5 STEP 0,35
------------------------
104 FOR T = P TO L STEP J
------------------------
134 FOR Z = B3 TO (C + D)/2 STEP 5 * J
```

Naturalmente i valori dei simboli e delle variabili dovranno essere definiti all'esterno dell'iterazione, o mediante istruzioni READ o INPUT o mediante calcoli.

2.5. LE VARIABILI INDICIZZATE

Problema 4

Riprendere l'archivio definito nel problema 2 e leggere le M schede che lo compongono. Determinare per ciascuno degli ordini la somma S = X + Y ed estrarre gli ordini tali che S < 3000.
Calcolare il totale T delle S < 3000.

Programma 4

```
 10 DATA 1, 25, 645, 2, 368, 552, 3, 465, 2250
 11 DATA 4, 236, 1248, 5, 695, 4250, 6, 182, 749
 12 DATA 7, 707, 961, 8, 747 4650, 9, 95, 1942
 13 DATA 10, 951, 2687, 11, 433, 1044, 12, 285, 700
 14 INPUT M
 15 DIM N (M), X(M), Y(M), S(M)
 20 FOR I = 1 TO M
 30 READ N (I), X(I), Y(I)
 40 NEXT I
 50 PRINT «ORDINI INFERIORI A 3000»
 60 PRINT
 70 PRINT «N-----X-----Y-----S»
 80 PRINT : PRINT
 90 T = 0
100 FOR I = 1 TO M
110 S = X (I), X(I), S(I)
120 IF S > = 3000 THEN 140
130 PRINT N(I), X(I), Y(I), S(I)
135 T = T + S
140 NEXT I
145 PRINT
150 PRINT «TOTALE =», T
160 END
```

Contrariamente a quanto si è fatto precedentemente, si è voluto questa volta leggere l'insieme dei dati dell'archivio prima di cominciare l'elaborazione. Si guadagna così in velocità di esecuzione.
È necessario prevedere per questo il posto in memoria centrale. Si ottiene questo scopo mediante l'istruzione DIM (abbreviazione di DIMENSION) Quando si scrive:

```
14 DIM N(12), X(12), Y(12), S(12)
```

Si riservano dei blocchi di 12 parole di memoria.
I simboli N,X,Y, S designano ciascuno un'area di memoria, e per individuare una posizione precisa nell'area si utilizza l'indice.
Nel nostro esempio, quando si incontra l'istruzione

```
30 READ N(I), X(I), Y(I)
```

il primo dato letto sarà collocato nella parola N(1), il secondo in X(1) e il terzo in Y(1). Il quarto dato va a finire in N(2)...
I simboli N(I), X(I) sono *variabili indicizzate*. Se l'indice non supera 10, l'istruzione DIM è inutile.

Si noti che l'indice, nell'istruzione DIM, può essere una lettera. In effetti, si è scritto:

```
15 DIM N(M), X(M), Y(M), S(M)
```

se si prende la precauzione di definire la variabile M preventivamente. Ad esempio, come qui, prevedendo l'istruzione:

```
14 INPUT M
```

che determina la stampa di un punto interrogativo ? e aspetta che sia immesso un numero tramite la tastiera.

Questa soluzione ha il vantaggio di permettere una maggiore flessibilità d'uso del programma. In effetti, quando si ha finito di battere le istruzioni DATA (archivio dei dati), si conosce il loro numero totale. Si può dunque fissare M. Nel nostro esempio sarà sufficiente battere 12 quando lo schermo mostra il primo punto d'interrogazione.
Tutte le operazioni valide per le variabili normali lo sono anche per quelle indicizzate. L'indice dev'essere un numero intero positivo o nullo. Dev'essere definito o calcolato in un'istruzione precedente. Se il calcolo dell'indice conduce a un valore decimale, viene conservata solo la parte intera.

Ad esempio, se:

```
30 K = 5/2
40 P(K) = 15,3
```

il valore 15,3 sarà assegnato a P(2).

— Ritorno sui comandi di stampa:
In questo esempio si può constatare che, mediante apici, l'impaginazione risulta semplificata.
L'istruzione 70 stamperà N, X, Y, ed S come titoli di colonne.
— Si può anche notare che possono essere scritte numerose istruzioni su una stessa linea se non si supera la sua capacità.
Sarà sufficiente separare le due istruzioni per mezzo di due punti.

Esempio:

```
80 PRINT : PRINT : PRINT
```

è equivalente a

```
80 PRINT
81 PRINT
82 PRINT          SALTO DI TRE LINEE
```

Esecuzione del programma 4

```
RUN
? 12

ORDINI INFERIORI A 3000

N        X        Y        S
1       25      645      670
2      368      552      920
3      465     2250     2717
4      236     1248     1484
6      182      729      931
7      707      961     1668
9       95     1842     1937
11     433     1044     1477
12     285      700      985

TOTALE = 12787

READY
```

Gli indici possono ugualmente essere espressioni aritmetiche; ad esempio:

50 P((K — 3)/4) = 160

questo non è valido per K = 3.

2.6. LE ITERAZIONI MULTIPLE. VARIABILI A PIÙ INDICI

Problema 5

Si vuole calcolare la serie di coefficienti C_n^k del binomio di Newton. Si ricordi che:

$$(a + b)^n = \sum_{k=1}^{n} C_n^k a^k b^{n-k}$$

$$\text{con } C_n^k = \frac{n!}{k!\,(n-k)!}$$

e la relazione ricorrente:

$$C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$$

Programma 5

```
 10 INPUT N
 20 DIM C(N, N + 1)
 30 C(1, 1) = 1
 40 C(1, 2) = 1
 50 FOR I = 2 TO N
 60 C(I, 1) = 1
 65 C(I, I + 1) = 1
 70 FOR J = 2 TO I
 75 C(I, J) = C(I — 1, J) + C(I — 1, J — 1)
 77 NEXT J
 80 NEXT I
 90 FOR I = 1 TO N
100 FOR J = 1 TO I + 1
110 PRINT C(I, J)
120 NEXT J
125 PRINT
130 NEXT I
140 STOP
150 END
```

Commenti

— Abbiamo utilizzato l'algoritmo del triangolo di Pascal: se si rappresentano i coefficienti dello sviluppo di $(a + b)$, $(a + b)^2$, $(a + b)^3$, ..., $(a + b)^n$ assegnando una linea a ciascun binomio si ottiene

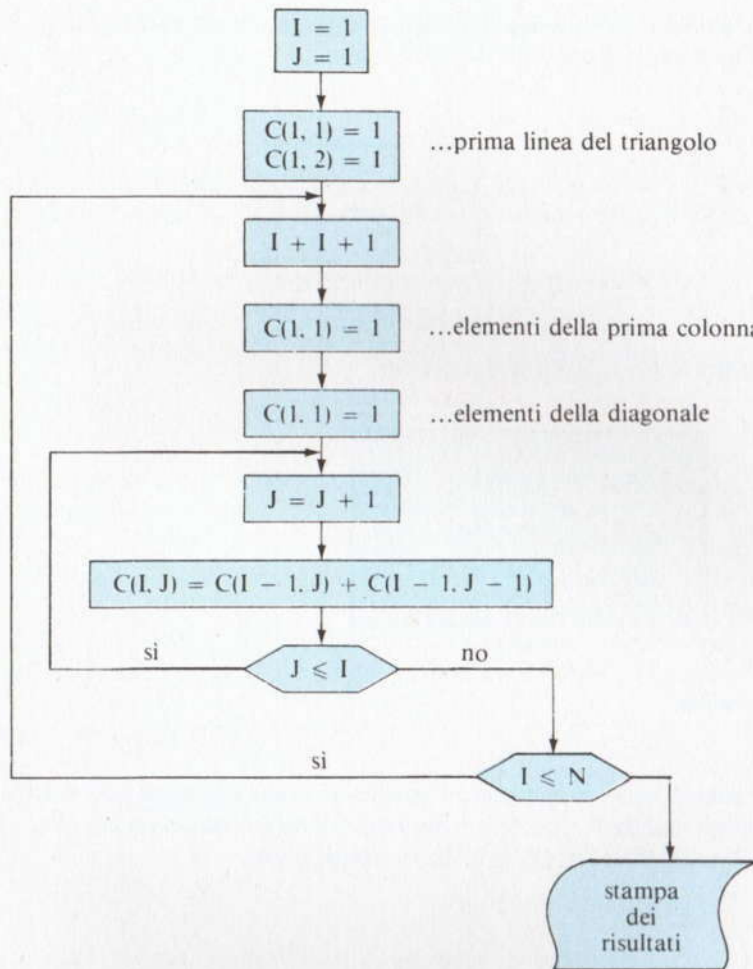
	j = 1	j = 2	j = 3		.... j = n
per i = 1 $a + b$ →	1	1			
per i = 2 $(a + b)^2$ →	1	2	1		
per i = 3 $(a + b)^3$ →	1	3	3	1	
....					
per i = n $(a + b)^n$ →	1	C_n^2	C_n^3	C_n^4	.... $C_n^n = 1$

Così disposti i coefficienti formano una matrice triangolare i cui elementi verificano la relazione ricorrente:

$$C(i, j) = C(i - 1, j) + C(i - 1, j - 1)$$

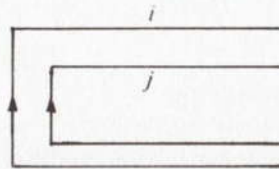
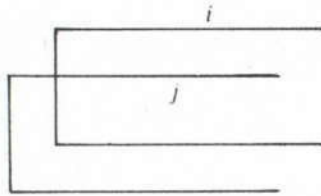
con le osservazioni seguenti:

— gli elementi della prima colonna sono tutti uguali ad 1
— lo stesso vale per gli elementi della diagonale.

Da qui si ottiene il diagramma di flusso seguente:

Ritorniamo al programma Basic:

— La linea 20 DIM C(N, N + 1) corrisponde alla presentazione di N × N + 1 parole di memoria dove saranno registrati i valori dei coefficienti calcolati. Evidentemente, anche qui N dovrà essere definito in un'istruzione precedente (istruzione 10).

— Il calcolo dei coefficienti si effettua nelle istruzioni 30 ÷ 85.

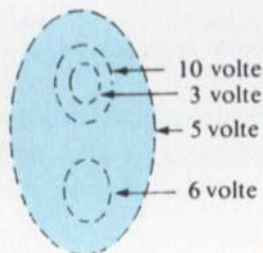
Questo calcolo utilizza delle iterazioni FOR incastrate, cioé una iterazione sull'indice J che si svolge per valori successivi dell'indice I, come indica lo schema seguente:

Attenzione: è possibile entrare in un'iterazione solo dall'inizio. È vietato lo scavalcamento del tipo qui rappresentato:

Ad esempio, il pezzo di programma:

```
200 FOR I = 1 TO 8
210 FOR J = 1 TO 5
220 NEXT I
230 NEXT J
```

non è valido.

— Le iterazioni di programma costituiscono lo strumento più importante della programmazione, poiché è nella ripetizione di una stessa sequenza di calcolo che il calcolatore mostra tutta la sua potenza.

Esercizi

1. Scrivere la forma generale delle istruzioni FOR e NEXT illustrate dallo schema seguente:

2. Calcolare i primi K numeri primi.
3. Eseguire il prodotto della matrice A(25, 25) con il vettore B(25).
4. Eseguire il prodotto di due matrici rettangolari A(25, 20), B(20, 30).

2.7. RIPETIZIONE DI UNA STESSA ISTRUZIONE. FUNZIONI

2.7.1. La funzione DEF FN

Capita spesso che uno stesso calcolo si ritrovi in differenti punti di uno stesso programma. È comodo allora disporre di un'istruzione che possa essere richiamata a volontà.
A questo fine è necessario *definire* la funzione e poterla richiamare.

— La definizione si ottiene mediante l'istruzione DEF FN.
— Il richiamo si ottiene mediante l'istruzione di assegnazione tradizionale.

Esempi

Ammettiamo di dover effettuare la conversione da gradi a radianti. Si potrà scrivere, ad esempio:

```
50 DEF FNR (D) = D * 3,14159/180
```

Ogni volta che si vorrà, si potrà ottenere la conversione per mezzo del richiamo della funzione FNR (...) nell'ambito dello stesso programma.

Ad esempio, scrivendo:

```
        70 T1 = FNR(37)
o       80 T2 = FNR(A + 5)
        ..................
o       90 PRINT FRN(14)
```

La forma generale dell'istruzione di definizione di funzione è:

n. linea DEF nome (lista degli argomenti) =
= espressione che racchiude, tra l'altro, questi argomenti.
Il nome richiede obbligatoriamente tre lettere, le cui prime due sono FN. Non si potranno mai definire più di 26 funzioni che sono FNA, FNB, ... FNZ.
L'espressione alla destra del segno = può essere una qualunque espressione aritmetica contenibile in una linea.

— Si possono dunque definire funzioni di numerose variabili, come la seguente:

```
100 DEF FNV(X,Y,Z) = X ↑ 2 + Y ↑ 2 + Z ↑ 2
```

che verrà richiamata con tre argomenti, quali:

```
120 U1 = FNV(4, —3, 2)
130 U2 = FNV (2.5, 1, 0)
```

2.7.2. Le funzioni biblioteca

Si può anche disporre, per effettuare dei calcoli comuni, di funzioni fornite da una biblioteca permanente. Ad esempio, ecco qui una lista che si ritrova su tutte le macchine, con varianti nella denominazione:

Sin (x)	Calcolo del seno dell'angolo x
Cos (x)	Calcolo del coseno dell'angolo x
Tan (x)	Calcolo della tangente dell'angolo x
Atg (x)	Calcolo dell'arcotangente dell'angolo x
Exp (x)	Calcolo dell'esponenziale di x
Abs (x)	Valore assoluto di x
Rqu (x)	Radice quadrata di x
Int (x)	Parte intera di x
Rnd (x)	Fornisce un numero aleatorio (probabilità uniforme)

Queste funzıoni hanno tutte una forma generale:

nome di 3 lettere seguito da un argomento

L'argomento può essere una costante, una variabile o una espressione. Si scrive ad esempio:

```
20 A = Sin (B + C)
30 R = Rqu (X2 + Y2)
40 P = Abs (X — Y) + Rqu (X Log Y)
```

Una funzione biblioteca può figurare in qualunque istruzione di calcolo o di stampa. In particolare si può definire una funzione istruzione che comprenda una o più funzioni biblioteca.

Esempio

```
50 DEF FNW = Rqu (A ↑ 2 + B ↑ 2 + X ↑ 2)
```

— Una funzione particolare: TAB(N).

Essa permette di posizionare i caratteri da visualizzare o da stampare (TAB significa tabulazione).
Il parametro N può essere una costante, una variabile o un'espressione. In tutti i casi è la parte intera del valore che viene presa in conto. Se si scrive PRINT TAB (4, 3) A il valore di A sarà visualizzato dopo 4 spazi.
Si può anche scrivere PRINT TAB (10 — FNY(X)); «0»
o ancora PRINT TAB (Y); «*»; TAB (Z); «0»

Quest'ultima istruzione permette di rappresentare le curve (Y) e f(Z).

Esercizi

1. Tracciare le funzioni Y = Sen (x/50 + 3x/1000) e Z = Cos (X/30 + 0,25) su uno stesso grafico.
2. Realizzare una funzione che permetta di allineare in colonne dei numeri decimali qualunque, in rapporto alla virgola.
Ad esempio, se si vogliono rappresentare 347,50 e 1253,839 si deve ottenere:

```
 347,50
1253,839
```

2.8. RIPETIZIONE DI UNO STESSO PROBLEMA. CONCETTO DI SOTTOPROGRAMMA

L'utilizzazione di una funzione come detto in precedenza è limitata ai casi in cui l'insieme dei calcoli può essere contenuto su una linea d'istruzione Basic. Spesso sarebbe comodo utilizzare lunghe sequenze di calcolo in diversi punti del programma, come nell'esempio che segue.

Problema 6

Riprendiamo l'archivio fatture definito nei problemi 2, 3 e 4. Determiniamo di nuovo la somma S = X + Y e selezioniamo gli ordini secondo tre categorie:

- piccoli ordini: tali che $S \leqslant S_1$
- *ordini medi: tali che* $S_1 < S \leqslant S_2$
- *grossi ordini: tali che* $S > S_2$

Si richiede la loro visualizzazione successiva secondo queste tre classi, con il totale parziale e il totale generale.

Programma 6

```
  5 DATA
 10 INPUT M
 20 DIM N(M), X(M), Y(M), S(M)
 30 FOR I = 1 TO M
 40 READ N(I), X(I), Y(I)
 50 NEXT I
 60 INPUT S1, S2
 70 FOR I = 1 TO M
 80 S(I) = X(I) + Y(I)
 85 NEXT I
 86 T = 0
 90 FOR K = 1 TO M
 95 IF S(K) > S1 THEN 140
 96 T = T + S(K)
```

```
110 PRINT «...N.....X.....Y.....S»
120 PRINT
130 PRINT N(K); X(K); S(K)
140 NEXT K
150 PRINT «TOTAL  = .......» T
160 T = 0
165 FOR K = 1 TO M
167 IF (S(K) — S1) * (S2 — S(K)) > = 0 THEN 200
170 T = T + S(K)
180 PRINT «...N.....X.....Y.....S»
190 PRINT N(K); X(K); Y(K); S(K)
195 PRINT
200 NEXT K
205 T = 0
210 FOR K = 1 TO M
220 IF S(K) — S2 < 0 THEN 270
230 T = T + S(K)
240 PRINT «...N.....X.....Y.....S»
250 PRINT
260 PRINT N(K), Y(K), S(K)
270 NEXT K
280 STOP
999 END
```

Programma 6bis

```
  5 DATA..........
 ................
 10 INPUT M
 20 DIM N(M), X(M), Y(M), S(M)
 30 FOR I = 1 TO M
 40 READ N(I), X(I), Y(I)
 45 S(I) = X(I) + Y(I)
 50 NEXT I
 60 FOR J = 1 TO 3
 70 INPUT S1, S2
 80 GOSUB 100
 90 NEXT J
 95 STOP
100 T = 0
102 PRINT «...N.....X......Y......S»
105 FOR K = 1 TO M
110 IF (S(K) — S1) * (S2 — S(K)) > 0 THEN 130
115 T = T + S(K)
120 PRINT N(K); X(K); Y(K); S(K)
```

```
130 NEXT K: PRINT
140 PRINT «TOTAL = ......» T
150 RETURN
999 END
```

Commenti

I programmi 7 e 7bis eseguono la stessa elaborazione, ma il secondo richiede 16 istruzioni Basic in meno. In effetti le sequenze (86 ÷ 150) e (160 ÷ 200) del programma 7 rappresentano le stesse operazioni o almeno qualcosa che si può ricondurre ad un'unica elaborazione rappresentata dalla sequenza (100 ÷ 140) del programma 7bis.
Si è dunque deciso di considerare quest'ultima sequenza come un sottoprogramma, cioé come un insieme di istruzioni che realizzano un'elaborazione completa e suscettibile di essere richiamata ad ogni momento in un programma definito principale.
Nel programma 7bis è la sequenza (10 ÷ 95) che rappresenta il programma principale. Vi si leggono i dati dell'archivio DATA, si realizzano le somme S(I), poi si fa appello per tre volte alla sequenza d'elaborazione che va da 100 a 150.
Per utilizzare un sottoprogramma è necessario poterlo richiamare, precisargli i valori dei parametri che gli sono necessari e, quando l'elaborazione è finita, ritornare al programma principale.
È per questo che si scrive

```
 80 GOSUB 100
150 RETURN
```

cioé andare ad eseguire le istruzioni a partire dalla linea 100 fino ad incontrare l'istruzione che segue immediatamente 80. Nel caso specifico si tratterà di 90 NEXT J, cioé

```
70 INPUT SI, S2
```

2.9. SOTTOPROGRAMMI INDIPENDENTI

L'utilizzo dell'istruzione di richiamo GOSUB permette di utilizzare numerose volte lo stesso sottoprogramma in un programma dato, ma questa sequenza dovrà far parte del programma principale. In particolare il programma principale e il sottoprogramma sono compilati contemporaneamente.
Esiste un altro tipo di sottoprogramma che permette d'essere compilato e memorizzato indipendentemente dal programma principale. La forma generale

di un tale sottoprogramma è:

```
      Nome (di 3 caratteri alfanumerici)
    10 ''
    20 ''
     • ''
     • ''
    n. di linea RETURN
```

Questo viene richiamato nel programma principale da:

N° linea CALL nome del sottoprogramma

Riprendiamo l'esempio 6bis; possiamo introdurvi il nuovo concetto nel modo seguente:

Programma 6ter

Programma principale

```
 10 DATA.....
 20 INPUT M
 30 DIM N(M), X(M), Y(M), S(M)
 40 FOR I = 1 TO M
 50 READ N(I), X(I), Y(I)
 60 S(I) = X(I) + Y(I)
 70 NEXT I
 80 FOR J = 1 TO 3
 90 INPUT S1, S2
 95 CALL TRI
 97 NEXT J
999 END
```

Sottoprogramma

```
TRI
10 T = 0
20 FOR K = 1 TO M
30 IF(S(K) — S1) * (S2 — S(K)) > 0 THEN 60
40 T = T + S(K)
50 PRINT N(K); X(K); S(K)
55 PRINT «N.....X......Y......S»
60 NEXT K : PRINT
70 PRINT «TOTAL = ......», T
80 RETURN
```

Commenti sul programma 6ter

L'istruzione GOSUB richiama un sottoprogramma incluso nel programma principale, e quindi compilato contemporaneamente a questo. L'istruzione

CALL permette al contrario di chiamare un sottoprogramma che non fa parte integrante del programma principale. In particolare il sottoprogramma può essere stato compilato separatamente e collocato in memoria sotto un nome particolare. Nel seguito potrà essere richiamato da differenti programmi.
Nel nostro caso, l'istruzione:

```
95 CALL TRI
```

richiama il sottoprogramma TRI assegnando alle variabili N,X,Y,S i valori che esse hanno nel programma principale al momento della chiamata.
Il ritorno al programma principale si effettua qui per mezzo dell'istruzione RETURN (ve ne possono essere numerose). Il ritorno si effettua sempre alla prima istruzione che segue l'istruzione CALL.
I numeri di linee del programma principale e del sottoprogramma sono naturalmente indipendenti.

2.10. ELABORAZIONE DI CARATTERI ALFANUMERICI. STRINGHE

Il Basic permette ugualmente di realizzare talune elaborazioni sulle variabili non numeriche, cioé sulle lettere e sui caratteri speciali, definite variabili alfanumeriche.
Per indicare che una variabile è alfanumerica, si fa seguire al nome della variabile il simbolo del dollaro $.
Tutte le operazioni di lettura, scrittura o assegnazione rimangono valide. Si potrà scrivere, ad esempio:

```
20 LET A$ = «MOLTE»
30 LET B$ = «GRAZIE»
40 PRINT B$; A$
50 END
```

Questo programma produrrà la stampa del testo:

```
GRAZIE MOLTE
```

La seconda parte d'una istruzione di assegnazione può essere una stringa di caratteri (e in questo caso è necessario metterli tra virgolette), come nell'esempio precedente, oppure una variabile alfanumerica. Questa conterrà anch'essa una stringa di caratteri che non potrà superare i 15 caratteri.

Esempio: 60 LET C$ = H$

Una variabile alfanumerica può essere indicizzata. Sarà necessario allora definirla a mezzo di un'istruzione DIM. È permesso un solo indice.

Esempio: 10 DIM A$ (20)

che prenoterà 20 volte 15 caratteri in memoria.

— Lettura e scrittura di stringhe di caratteri:
READ e INPUT possono comportare delle variabili alfanumeriche contemporaneamente alle variabili numeriche.
Ad esempio, si può scrivere:

```
10 READ X$, Y$, M, P
20 INPUT N$, N
30 DATA «VENERDI», «13 MARZO 1980», 10, 15
```

Si noti che i dati alfanumerici sono messi tra virgolette. Queste sono indispensabili solamente quando la stringa di caratteri comincia con una cifra o contiene una virgola. Si possono utilizzare sistematicamente in caso di dubbio.
L'istruzione PRINT si utilizza anch'essa come per le variabili numeriche.

— Operazioni di confronto sulle variabili alfanumeriche. Si può utilizzare l'istruzione IF...THEN che confronta due stringhe di caratteri, uno ad uno sul loro codice BCD (codice decimale binario). Gli spazi bianchi sono ignorati.

Esempi

```
 10 IF A$ = «SMITH» THEN 240
120 IF B$ < > C$ THEN 300
410 IF «APRIL» < = M$ THEN 500
```

CAPITOLO 3

LE ESTENSIONI DEL BASIC

I compilatori Basic utilizzati sui minicalcolatori offrono possibilità molto maggiori delle funzioni di base del linguaggio. In questo capitolo passeremo in rassegna gli aspetti più interessanti del Basic Plus della Digital Equipment Corporation, utilizzato sui calcolatori PDP della serie 11 che funzionano con il sistema operativo RSTS (Sistema a Ripartizione di Tempo).

3.1. LE VARIABILI

In Basic esistono tre tipi di variabili:

— variabili reali
— variabili intere(*)
— variabili stringhe di caratteri

Nel modo standard (modo NOEXTEND implicito) il nome di questi tre tipi di variabili è formato da uno o due caratteri di cui il primo è necessariamente una lettera e il secondo (facoltativo) una cifra. La distinzione tra i tre tipi si realizza mediante un segno che segue il nome. I nomi delle variabili intere sono seguite dal segno %; le variabili stringhe di caratteri sono seguite dal segno $.

Esempio

```
10      A$  = «ABC»  \ Z3$  = «DEF» \ 1$ = «GHI»
20      A%  = 10%    \ Z3%  = 5     \ 1% = 3%
30      A   = 10,234 \ Z3   = 2     \ 1  = —2,342
40      PRINT A$, Z3$, X1$
\       PRINT A%, Z3%, X1%
\       PRINT A, Z3, X1
32767   END
Ready
```

(*) Le variabili intere sono memorizzate su due byte; possono quindi essere rappresentate sotto questa forma solo le variabili i cui valori sono compresi tra — 32 768 e 32 767.

```
RUNNH
ABC          DEF          GHI
 10           5             3
 10,234       2           —2,342
Ready
```

È anche possibile utilizzare il modo EXTEND. In questo caso i nomi delle variabili possono contenere fino a 30 caratteri. Per indicare al compilatore l'utilizzazione di questo modo, è necessario precisarlo nella prima istruzione del programma.

Esempio

```
 5   EXTEND
10   VARIABILE.INTERA% = 5%
20   REALE = 123, 23
30   STRINGA.CARATTERI$ = «ABCDEFG»
40   PRINT VARIABILE.INTERA%, REALE, STRINGA.CARAT-
     TERI$
Ready
```

```
RUNNH
 5    123,23       ABCDEFG
Ready
```

3.2. OPERAZIONI SULLE STRINGHE DI CARATTERI

Una stringa di caratteri è costituita da una sequenza di caratteri alfanumerici (definiti in codice ASCII) considerati come un insieme.

Esempio

```
10     A$ = «ESEMPIO»
Ready
```

La lunghezza d'una variabile che rappresenta una stringa di caratteri non è limitata. Queste variabili, come quelle relative ai numeri reali e interi, possono essere indicizzate.

Esempio

```
10     DIM E8$ (2,7), A$ (13%)
Ready
```

Il Basic Plus comprende un certo numero di funzioni intrinseche che definiscono delle operazioni sulle variabili «stringhe caratteri»; citeremo solamente le più utilizzate.
Concatenazione di numerose stringhe di caratteri.

Esempio

```
10      A$ = «ABC» / B$ = «DEF»
20      C$ = A$ + B$ / D$ = B$ + «X YZ»
30      PRINT A$ PRINT B$ PRINT C$, D$
Ready
```

ASCII (A$)

Valore ASCII del primo carattere della stringa.

Esempio

```
10      INPUT «STRINGA DI CARATTERI»; A7$
20      PRINT «VALORE ASCII =»; ASCII (A7$)
Ready

RUNNH
STRINGA CARATTERI? PROVA
VALORE ASCII = 69
Ready
```

CHR$(N%)

Generazione d'un carattere rappresentato dal carattere ASCII N.

Esempio

```
10      INPUT «NR CARATTERE ASCII»; N%
20      PRINT N%; «RAPPRESENTA IL CARATTERE»;
        CHR$ (N%); «.»
Ready

RUNNH
NR CARATTERE ASCII? 65
65 RAPPRESENTA IL CARATTERE «A»
Ready
```

Questa funzione è spesso utilizzata per far funzionare la suoneria del terminale.

```
10        PRINT CHR(7%);
Ready
```

STRING$ (N1%, N2%)
Generazione d'una stringa di N1% caratteri rappresentati dal carattere ASCII N2.

Esempio

```
10        INPUT «LUNGHEZZA»; N1%
20        INPUT «NR CARATTERE ASCII»; N2%
30        PRINT STRING$ (N1%, N2%)
Ready

RUNNH
LUNGHEZZA? 15
NR CARATTERE ASCII? 45
---------------------
Ready
```

Lo si utilizza nelle stampe per generare le cornici delle tabelle, per sottolineare i titoli eccetera.

LEFT (A$, N%)
Selezione d'una sottostringa della stringa di caratteri A$, su N caratteri partendo da sinistra.

Esempio

```
10        A$ = «ABCDEF» / B$ = «XYZ»
20        PRINT LEFT (A$, 4%) / PRINT LEFT (B$, 4%)
/         PRINT LEFT (A$, 3%) + LEFT (B$, 1%)
Ready

RUNNH
ABCD
XYZ
ABCX
Ready
```

RIGHT (A$, N%)
Selezione d'una sottostringa della stringa di caratteri A$, a partire dall'*n*esimo carattere, verso destra.

Esempio

```
10      A$ = «ABCDEF» / B$ = «XYZ»
20      PRINT RIGHT (A$, 4%) / PRINT RIGHT (B$, 4%)
/       PRINT RIGHT (A$, 2%) + RIGHT (B$, 3%)
Ready

RUNNH
DEF
BCDEFGZ
Ready
```

MID (A$, N1%; N2%)
Selezione d'una sottostringa della catena di caratteri A a partire dall'*n*1esimo carattere per N2 caratteri.

Esempio

```
10      A$ = «ABCDEF» / B$ = «XYZ»
20      PRINT MID (A$, 2%, 3%)
/       PRINT MID (A$, 3%, 2%)
/       PRINT MID (A$, 5%, 5%)
/       PRINT MID (A$, 1%, 2%) + MID (B$, 2%, 2%)
Ready


RUNNH
BCD
CD
ABYZ
Ready
```

LEN (A$)
Determinazione della lunghezza d'una stringa di caratteri.

Esempio

```
10      INPUT A$
20      PRINT LEN (A$)
30      GOTO 10
Ready
```

```
RUNNH
? PROVA
5
? BASIC-PLUS
10
Ready
```

INSTR (N1, A$, B$)
Ricerca, a partire dall'N1-esimo carattere, della posizione del primo carattere della sottostringa B$ nella stringa A$.
Se B$ è una stringa vuota, il valore restituito è uguale a 0, se la sottostringa B$ non figura in A$ il valore restituito è 1.

Esempio

```
10      A$ = «ABCDEFGHIJ» / B$ = «GH» / C$ = «XY» D$ = «»
20      PRINT INSTR (1, A$, B$)
/       PRINT INSTR (8, A$, B$)
/       PRINT INSTR (1, A$, B$)
/       PRINT INSTR (1, A$, C$)
/       PRINT INSTR (1, A$, D$)
Ready

RUNNH
7
0
0
1
Ready
```

VAL (A$)
Valore di una stringa di caratteri numerici. Se la stringa A$ non è numerica, il sistema segnala l'errore e restituisce il valore 0.

Esempio

```
10      A$ = «312345» / B$ = «1234,567»
20      X% = VAL (A$) / X = VAL (B$)
30      PRINT X%, X
Ready

RUNNH
12345           1234,57
Ready
```

NUM$ (N)
Rappresentazione d'un numero sotto forma di stringa di caratteri. La lunghezza della stringa risultante sarà uguale al numero di cifre del numero (più eventualmente una o due posizioni supplementari per la virgola e il segno se il numero è negativo) aumentata di due posizioni corrispondenti agli spazi generati, l'uno davanti, l'altro dietro il numero.

Esempio

```
10      N = 1234,56
20      A$ = NUM$ (N)
30      PRINT «*»; A$; «*»
Ready

RUNNH
* 1234,56 *
Ready
```

NUM1$ (N)
Funzione equivalente a NUM$, ma che non genera gli spazi.

Esempio

```
10      N = 1234,56
20      A$ = NUM1$ (N)
30      PRINT «*»; A$; «*»
Ready

RUNNH
*1234,56*
Ready
```

3.3. LE INTERRUZIONI DI SEQUENZA

La scelta di ordini condizionali è molto vasta nel Basic Plus. Oltre all'istruzione IF....THEN... già studiata, si trovano infatti le seguenti istruzioni:

ON	espressione	GOTO n. linea 1, n. linea 2.....
ON	espressione	GOSUB n. linea 1, n. linea 2.....
IF	condizione	THEN istruzione ELSE istruzione
	istruzione	IF condizione

La prima istruzione permette di saltare a linee diverse in funzione del valore d'una variabile o d'una espressione.

La seconda, invece di eseguire un salto, esegue un sottoprogramma. La terza istruzione permette d'intraprendere azioni appropriate qualunque sia il risultato del test. È possibile inoltre mettere numerose istruzioni al seguito dell'ELSE (al limite anche un'ulteriore istruzione IF...THEN...ELSE); al contrario THEN non potrà che essere seguita da una sola istruzione (diversa da IF).

L'ultima istruzione è una forma particolare di salto condizionale. In effetti, in funzione della condizione che segue l'IF, si esegue o meno l'istruzione che la precede, altrimenti si esegue un salto all'istruzione immediatamente seguente.

Esempio 1

```
10        INPUT «VALORE DI X»; X
20        ON X GOTO 100, 110,120
100       PRINT «LINEA 100 X =»; X / GOTO 10
110       PRINT «LINEA 110 X =»; X / GOTO 10
120       PRINT «LINEA 120 X =»; X / GOTO 10
32767     END
Ready

RUNNH
VALORE DI X? 1
LINEA 100 X = 1
VALORE DI X? 2
LINEA 110 X = 2
VALORE DI X? 3
LINEA 120 X = 3
VALORE DI X? 4
? ON statement out of range at line 20 (fuori limiti all'istruzione 20)
Ready
```

Esempio 2

```
10        INPUT «VALORE DI X»; X
20        ON X GOSUB 100,110,120
30        GOTO 10
100       PRINT «LINEA 100 X =»; X / RETURN
110       PRINT «LINEA 110 X =»; X / RETURN
120       PRINT «LINEA 120 X =»; X / RETURN
32767     END
Ready

RUNNH
VALORE DI X? 1
LINEA 100 X = 1
VALORE DI X? 2
```

LINEA 110 X = 2
VALORE DI X? 3
LINEA 120 X = 3
VALORE DI X? 0
? ON statement out of range at line 20 (fuori limiti all'istruzione 20)

Ready

Esempio 3

In funzione del n. del giorno, del mese e dell'anno si vuole calcolare il giorno della settimana.
Sia I il n. del giorno, M del mese ed Y dell'anno.
L'algoritmo utilizzato è il seguente (*):

$$\text{n. giorno della settimana} = \text{modulo } 7(I - 1 + \text{INT}(5Y1/4) - \text{INT}(Y1/100) + \text{INT}(Y1/400) + \text{INT}(13(1 + M1)/5))$$

dove

$Y1 = Y - 1$ ed $M1 = M + 12$ se $M = 2$
$Y1 = Y$ ed $M1 = M$ negli altri casi

```
10      DIM J$ (6%)
20      FOR I% = 0% TO 6%
/       READ J$ (I%)
/       NEXT I%
30      DATA DOMENICA, LUNEDÌ, MARTEDÌ, MERCOLEDÌ, GIO-
        VEDÌ, VENERDÌ, SABATO
35      REM IMMISSIONE E CONTROLLO DELLA DATA
40      INPUT «NUMERO DEL GIORNO»; J%
/       GOTO 40 IF J% < 1% OR J% > 31%
50      INPUT «N. DEL MESE»; M%
/       GOTO 50 IF M% < 1% OR M% > 12%
60      IF (M% = 4% OR M% = 6% OR M% = 9% OR M% = 11%)
        AND J% > 30% THEN 40 ELSE IF M% = 2% AND J% > 29%
        THEN 40
70      INPUT «ANNO»; Y%
/       GOTO 70 IF Y% < 1%
80      IF Y/4 — INT(Y/4) < > 0 THEN IF M% = 2% AND J% > 28%
        THEN 40
85      REM ALGORITMO
90      M1% = M%
/       Y1% = Y%
```

(*) Si considera l'inizio dell'anno al 1° marzo; i mesi di gennaio e febbraio sono dunque il 13° e 14° mese dell'anno precedente (il mese di settembre è, come l'indica il suo nome, il 7° dell'anno), INT significa parte intera. Modulo è il resto della divisione intera di due variabili: per esempio Modulo 7(22) = 1.

```
/        IF M1% < = 2% THEN M1% = M1% + 12
                              / Y1% = Y1% — 1%
100      J1% = J% — 1 + INT(5 * Y1% / 4%) — INT(Y1% / 100%) +
         INT(Y1% / 400%) + INT(13 * (1 + M1%)/ 5%)
110      J1% = J1% — INT(J1% / 7%) * 7% ! CALCOLO DEL MODU-
         LO 7
120      PRINT «IL»; J%; «/»; M%; «/»; Y%; «È UN»; J$ (J1%)
32767    END
Ready

RUNNH
NUMERO DEL GIORNO? 1
N. DEL MESE? 1
ANNO? 1980
IL 1/1/1980 È UN MARTEDÌ
Ready
```

3.4. LE ITERAZIONI DI PROGRAMMA

Oltre all'istruzione classica

FOR variabile = espressione TO espressione STEP espressione

si trovano in Basic altre possibilità di costruzione di cicli iterativi, e precisamente:

FOR variabile = espressione STEP espressione WHILE condizione
FOR variabile = espressione STEP espressione UNTIL condizione

Inoltre esiste una forma speciale d'iterazione di programma per la quale si utilizza una sola istruzione di programma.

Esempio 1

```
10       FOR X = 1 STEP 2 WHILE Z < 30
20       Z = X ** 2
30       PRINT X, Z
40       NEXT X
32767    END
Ready
```

```
RUNNH

1       1
3       9
5      25
7      49

Ready
```

In questo esempio saranno eseguite le linee 20 e 30 fintanto che la condizione è vera.

Esempio 2

```
10       FOR X = 1 STEP 2 UNTIL Z > = 30
20       Z = X ** 2
30       PRINT X,Z
40       NEXT X
32767    END

Ready
```

Questo esempio si differenzia da quello precedente solo per la linea 10. Le linee 20 e 30 saranno eseguite fino a che questa condizione risulta vera.
Le variazioni WHILE e UNTIL sono utilizzate quando l'uscita dall'iterazione dipende da una variabile diversa dall'indice.
Le iterazioni che non riguardano una sola istruzione hanno la forma seguente:

istruzione FOR variabile = espressione TO espressione STEP espressione
istruzione FOR variabile = espressione STEP espressione Y UNTIL / WHILE (condizione)

istruzione UNTIL condizione
istruzione WHILE condizione

3.5. GLI OPERATORI LOGICI

Gli operatori logici sono utilizzati in talune operazioni logiche sulle variabili intere, ma soprattutto nelle istruzioni IF-THEN. Nel Basic Plus esistono 6 operatori logici:

NOT	negazione logica
AND	prodotto logico
OR	somma logica
XOR	disgiunzione logica
IMP	implicazione logica
EQV	equivalenza logica

In pratica gli operatori più utilizzati sono AND e OR.
Le due tabelle seguenti riassumono le operazioni che è possibile effettuare con questi operatori (*).

A	NOT A
V	F
F	V

A	B	A AND B	A OR B	A XOR B	A IMP B	A AQV B
V	V	V	V	F	V	V
V	F	F	V	V	F	F
F	V	F	V	V	V	F
F	F	F	F	F	V	V

Esempio

```
10      INPUT «IMMETTETE TRE NUMERI IN ORDINE CRESCEN-
        TE»; A,B,C
20      IF A < B AND B < C THEN PRINT «GRAZIE»
        ELSE PRINT «RICOMINCIARE» / GOTO 10
32767   END
Ready

RUNNH
IMMETTETE TRE NUMERI IN ORDINE CRESCENTE? 1,3,2
RICOMINCIATE
IMMETTETE TRE NUMERI IN ORDINE CRESCENTE? 5,4,3
RICOMINCIATE
IMMETTETE TRE NUMERI IN ORDINE CRESCENTE? 1,2,3
GRAZIE
Ready
```

Al momento dell'utilizzazione degli operatori logici sulle variabili intere, le operazioni logiche sono effettuate sui bit contenuti nelle parole.

Esempio

Questo piccolo programma è molto utile quando è necessario verificare se un numero è pari o dispari.

```
10      INPUT «IMMETTERE UN NUMERO INTERO»; I%
20      IF (I% AND 1%) = 1% THEN PRINT I%; «È DISPARI»
        ELSE PRINT I%; «È PARI»
32767   END
Ready
```

(*) Le espressioni logiche possono assumere unicamente due valori, vero (V) o falso (F). Per le operazioni sulle variabili intere risulta conveniente sostituire V con 1 ed F con 0.

```
RUNNH
IMMETTERE UN NUMERO INTERO? 123
123 È DISPARI
Ready

RUNNH
IMMETTERE UN NUMERO INTERO? 4
4 È PARI
Ready
```

3.6. SOTTOPROGRAMMI DI CORREZIONE (PROGRAMMABILI) DEGLI ERRORI

Nel corso dell'esecuzione di un programma il sistema può incontrare degli errori (per esempio divisione per zero, immissione di un numero sbagliato a seguito di una operazione di INPUT ecc.).
Abitualmente, in caso di errore, il sistema stampa un messaggio di errore e arresta l'esecuzione del programma. Il Basic Plus offre la possibilità di correggere certi errori senza arrestare l'esecuzione del programma, utilizzando un sottoprogramma particolare.
A questo fine tutti gli errori sono numerati e si dispone di due variabili riservate ERR ed ERL che contengono, quando si produce un errore, il numero dell'errore e quello dell'istruzione (linea) in cui questo si verifica.
Per poter utilizzare il sottoprogramma di correzione errori è necessario segnalarlo al sistema mediante istruzione

ON ERROR GOTO n. di linea

che dev'essere collocata prima di qualunque istruzione in grado di richiamare questo sottoprogramma.
Il sottoprogramma di correzione errori comincia dalla linea in cui il numero è specificato nell'istruzione. In questo sottoprogramma si utilizza spesso l'istruzione

RESUME n. di linea

Questa istruzione permette di uscire dal sottoprogramma di errore e di riprendere l'esecuzione del programma dalla linea specificata nell'istruzione.

Esempio

```
5     ON ERROR GOTO 32700
10    PRINT «IMMETTERE UN NUMERO INTERO»
20    INPUT J%
30    PRINT «GRAZIE» / GOTO 32767
```

```
32700    IF ERR = 50 AND ERL = 20 THEN PRINT «UN NU-
         MERO INTERO, PREGO»
/        RESUME
32767    END
Ready

RUNNH
IMMETTERE UN NUMERO INTERO? ABCD
UN NUMERO INTERO, PREGO
? 1,234
UN NUMERO INTERO, PREGO
? 5
GRAZIE
Ready
```

3.7.COMPLEMENTI ALLE ISTRUZIONI UTILI

Citeremo qui alcune istruzioni di diversa natura ma accomunate da una frequente utilizzazione.

INPUT LINE variabile stringa di caratteri

Questa istruzione, simile all'istruzione INPUT, permette l'ingresso di una sequenza di caratteri da un terminale (ma anche da un archivio; vedi cap. 4, *Gli archivi*). La fine della stringa è segnalata da RETURN (ritorno carrello) (caratteri ASCII 13 e 10), questi due caratteri facendo ugualmente parte di questa stringa.

Esempio

```
10       PRINT «IMMETTERE UN TESTO»;
20       INPUT LINE A$
30       PRINT A$ / PRINT «LUNGHEZZA TESTO = »; LEN
         (A$)
32767    END
Ready

RUNNH
IMMETTERE UN TESTO? PROVA
PROVA
LUNGHEZZA TESTO = 7
Ready
```

Il Basic Plus offre la possibilità di definire delle funzioni su numerose linee. La struttura della sequenza delle istruzioni è la seguente:

DEF FNidentificatore lista di argomenti - definizione della funzione
FNEND

Le funzioni e i loro argomenti possono essere di qualunque tipo.

Esempio

```
10      DEF FNF(R%)
20      IF R% = 0% THEN FNF = 1% / GOTO 40
30      IF R% = 1% THEN FNF = 1% ELSE FNF = R% *
        (R% — 1%)
40      FNEND
100     INPUT «IMMETTERE UN NUMERO INTERO»; R%
110     PRINT «IL FATTORIALE»; R%; «È UGUALE A»;
        FNF (R%)
32767   END
Ready

RUNNH
IMMETTERE UN NUMERO INTERO? 6
IL FATTORIALE DI 6 È UGUALE A 30
Ready
```

Esempio

```
10      DEF FNC (A$, B$)
20      IF A$ < = B$ THEN C$ = A$ + «XXX» + B$ ELSE
        C$ = A$ + «YYY» + B$
30      FNC% = LEN(C$)
40      FNEND
100     INPUT «X$ =»; X$
110     INPUT «Y$ =»; Y$
120     PRINT FNC% (X$, Y$), C$
32767   END
Ready
RUNNH
X$ = ? A
Y$ = ? B
5      AXXXB
Ready
```

Quando i programmi diventano troppo grossi o certe parti di elaborazioni sono identiche in più programmi è possibile effettuare un salto verso un altro

programma. Questa azione è realizzata a mezzo della istruzione CHAIN (nome del programma) n. di linea.

Esempio

```
17000    CHAIN «PARTENZA» 31000
Ready
```

Arrivati sulla linea 17000 il programma passerà la mano alla linea 31000 del programma PARTENZA.

CAPITOLO 4

GLI ARCHIVI

4.1. CONCETTO DI ARCHIVIO

Il concetto di archivio è fondamentale e compare in tutte le applicazioni di gestione dati.
Per archivio s'intende una raccolta organizzata di dati che presentano un legame tra loro e che possono essere consultati individualmente in modo ripetitivo e sistematico.
Un archivio è dunque non solamente un insieme di dati utilizzati da un programma, ma ugualmente il programma stesso.
Un archivio, costituito da una serie di registrazioni, è memorizzato su un supporto fisico (disco magnetico, dischetto, nastro magnetico, schede o nastro perforato ecc.). Quando si parla di registrazione è necessario distinguere tra registrazione logica (o articolo) e registrazione fisica.
La registrazione logica è l'insieme delle informazioni relative alle entità elementari oggetto di un'elaborazione individuale nel corso di un processo iterativo cui è sottoposto l'archivio.
Un archivio è identificato da un certo numero di nomi; ad esempio nel Basic Plus tutto l'archivio è identificato da un nome (*) (di 6 caratteri al massimo), dal suffisso ed eventualmente dalla designazione del terminale su cui si trova.
Il suffisso è costituito da un punto seguito da 1-3 caratteri alfanumerici ed è attribuito dal sistema (ad esempio i programmi in Basic Plus hanno sempre il suffisso BAS, mentre quelli compilati in Basic hanno il suffisso BAC) oppure scelto dall'utilizzatore.
Per il Basic Plus esistono tre tipi di archivi:

— sotto forma ASCII
— in ingresso-uscita per registrazione
— in memoria virtuale.

4.2. APERTURA E CHIUSURA DI UN ARCHIVIO

Prima di eseguire un'operazione qualsiasi su un archivio, qualunque sia il suo tipo, è necessario aprirlo, cioé assegnargli un canale di entrata-uscita (da 1 a 12). Nel corso dell'esecuzione del programma ciascun riferimento a un archi-

---

(*) In realtà solo gli archivi memorizzati sui terminali periferici ad archivi strutturati - dischi, nastri magnetici - sono identificati da un nome. Gli archivi scritti sui terminali tipo stampante, lettore, perforatore di nastro, lettore/perforatore di schede, sono indirizzati specificando il solo terminale.

vio di dati si compie mediante il suo numero di riferimento.
L istruzione d'apertura di un archivio si scrive nella forma:

```
OPEN Nome archi-  ┌              ┐
     vio          │FOR INPUT     │AS FILE N. [RECORDSIZE espres-
                  │              │                       sione]
     Designazione │FOR OUTPUT    │
     del terminale└              ┘
```

```
100     OPEN «TOTO.DAT» AS FILE 2%
110     X% = 3%
/       OPEN «STOCK.FIC» AS FILE X%
120     A = «DRO:COMPTA. LST»
/       OPEN A FOR OUTPUT AS FILE 4% - Archivio su disco DRO
130     OPEN «LP:» AS FILE 5% - Apertura stampante
```

Le opzioni FOR INPUT e FOR OUTPUT permettono di aprire un archivio esistente o nuovo.
L'opzione FOR INPUT determina la ricerca dell'archivio specificato da parte del sistema.
Se trova l'archivio lo apre, altrimenti il sistema segnala l'errore «archivio assente» (CAN'T FIND FILE OR ACCOUNT).
Con l'opzione FOR OUTPUT l'archivio specificato viene aperto, sia che esso esista o meno. Se l'archivio esiste già, il suo contenuto viene distrutto prima dell'apertura. L'omissione dell'opzione provoca l'apertura dell'archivio in tutti i casi, senza alterazione del suo contenuto.
Tutti gli scambi di informazioni tra l'archivio e il programma avvengono tramite un'area riservata dal sistema nella memoria centrale e propria a ciascun canale d'ingresso-uscita.
Quest'area è definita area di trasferimento o tampone (*buffer* in inglese) e corrisponde alla dimensione di una registrazione fisica.
L'opzione RECORDSIZE permette di fissare la dimensione di quest'area. In ogni caso, quando l'opzione RECORDSIZE è omessa, il sistema assegna alle aree di trasferimento dati delle dimensioni che dipendono dal tipo di terminale. La dimensione del «buffer» associato dal sistema a un disco o a un lettore di nastro magnetico è di 512 caratteri.
Questa opzione viene generalmente utilizzata quando la dimensione della registrazione logica è superiore alla registrazione fisica. Non appena lo scambio dei dati tra il programma e l'archivio si è concluso, quest'ultimo dovrà essere chiuso. La chiusura interrompe i legami logici tra l'archivio e il canale che gli è stato associato. Quando un archivio è utilizzato in uscita, la chiusura ha anche l'effetto di provocare la scrittura del contenuto dell'area di comunicazione (buffer).

La forma generale di questa istruzione è la seguente:

CLOSE espressione, espressione......

dove l'espressione è un intero compreso tra 1 e 12.

Esempio

```
1000    CLOSE 1%, 7% ! Chiusura canale 1 e 7
1100    CLOSE I% FOR I% = 1% TO 12% ! Chiusura di tutti i canali
Ready
```

4.3. GLI ARCHIVI SOTTO FORMA ASCII

È la forma più semplice di organizzazione degli archivi. La scrittura e la lettura non sono possibili se non nell'ordine sequenziale (*).
La scrittura degli archivi ASCII è effettuata per mezzo dell'istruzione:

PRINT espressione, lista

L'espressione rappresenta il numero del canale di entrata-uscita. La lista contiene le variabili, le costanti e le espressioni da scrivere. Queste variabili, costanti ed espressioni sono separate da segni di punteggiatura (; o,) che definiscono la forma di scrittura e sono rappresentate, qualunque sia la loro natura, sotto forma di caratteri. Ciò richiede la trasformazione dei numeri interi e reali in stringhe di caratteri: questa funzione è svolta automaticamente dal sistema.
Nell'archivio ASCII gli articoli (o registrazioni logiche) sono separate tra loro dal carattere RETURN (ritorno carrello: caratteri ASCII 13 e 10). Un articolo può essere scritto con una o più istruzioni PRINT; il carattere di ritorno carrello è generato dall'istruzione PRINT nella quale l'ultimo elemento della lista non è seguito dal segno di punteggiatura (; o,).

Esempio

```
110     PRINT A$;
120     PRINT B$        !Scrittura di un articolo con due PRINT
130     PRINT A$; C$    !Scrittura di un solo articolo
140     PRINT B$        !Scrittura di un articolo
Ready
```

(*) La scrittura può essere effettuata su tutti i supporti (magnetici, perforatori di banda, stampante, console di visualizzazione). La lettura, al contrario, non è possibile se non su certi tipi di supporto (nastro magnetico, lettore di schede, nastro, tastiera). Nel testo che segue faremo l'ipotesi che l'archivio sia su un supporto ad accesso casuale (disco).

La lettura degli archivi sotto forma ASCII si effettua mediante l'istruzione:

INPUT LINE # (espressione), (variabile stringa di caratteri)

dove l'espressione rappresenta, come nell'istruzione precedente, il numero di canale.
Questa istruzione legge l'archivio fintanto che non incontra un RETURN. Il risultato di questa lettura è organizzato nella variabile stringa di caratteri.
Così numerose variabili o costanti scritte con una o con numerose istruzioni PRINT possono essere lette con una sola istruzione INPUT LINE.

Esempio

```
 90     ! SCRITTURA D'UN ARCHIVIO ASCII
100     OPEN «DBO:PROVA.DAT» AS FILE 1%
        ! APERTURA DELL'ARCHIVIO SU DISCO DBO
110     A$ = «PROVA»
120     FOR I% = 1% TO 3%
130     PRINT # 1%, «NO»; I%; 100% + I%; 1000% + I%
140     PRINT # 1, A$
150     NEXT I%
160     CLOSE 1%!Chiusura dell'archivio - scrittura ultimo blocco
        !
        ! LETTURA DELL'ARCHIVIO
        !
170     OPEN «DBO:PROVA. DAT» AS FILE 1%!Riapertura dell'ar-
        chivio
180     FOR I% = 1% TO 6%
190     INPUT LINE # 1, A$
200     PRINT A$
210     NEXT I%
32767   END
Ready
```

In questo esempio dopo l'apertura dell'archivio PROVA.DAT sul disco DBO si scrivono, per ciascuna iterazione di programma, due articoli. Il primo contiene quattro elementi (la costante «NO» e tre variabili: I%, I% + 10% e I + 1000%) sulla linea 130; il secondo non ne contiene che uno solo sulla linea 140.
Infine, dopo aver chiuso l'archivio per poter scrivere il contenuto dell'area di comunicazione, lo si riapre e si procede alla lettura e alla stampa su terminale delle registrazioni scritte precedentemente.

Esempio

```
        RUNNH
        NO 1      101      1001
        PROVA
        NO 2      102      1002
        PROVA
        NO 3      103      1003
        PROVA
Ready
```

Il vantaggio principale degli archivi sotto forma ASCII è la facilità d'impiego, l'inconveniente maggiore è invece l'organizzazione sequenziale.

4.4. GLI ARCHIVI D'INGRESSO-USCITA PER REGISTRAZIONI

Gli archivi ad ingresso-uscita costituiscono la forma più flessibile, ma al tempo stesso più complessa, per l'utilizzazione degli archivi in Basic Plus.
L'accesso agli archivi I/O per registrazioni può essere realizzato secondo l'ordine sequenziale, o direttamente mediante la posizione della registrazione. Si può mescolare all'interno di una registrazione qualunque tipo di variabile.

4.4.1. Lettura e scrittura d'un archivio: istruzioni GET e PUT

Le istruzioni GET e PUT permettono di effettuare gli scambi d'informazioni tra l'archivio e l'area di scambio ad esso associata. L'istruzione GET, il cui compito è di leggere e trasferire le informazioni dall'archivio verso il buffer, ha la forma seguente:

GET # (espressione) [,RECORD espressione 2]

La prima espressione indica il n. di canale, la seconda il n. di registrazione fisica che deve essere letta.
Quando l'opzione RECORD è assente, la lettura dell'archivio viene condotta nell'ordine sequenziale, cioé a ciascuna istruzione GET il sistema passa la registrazione fisica seguente nel buffer.
Se questa opzione è presente, essa permette l'accesso a una qualunque registrazione dell'archivio.

In effetti i blocchi fisici dell'archivio (aventi ciascuno la lunghezza di 512 caratteri) sono numerati da 1 ad N e l'espressione dell'opzione RECORD indica quale blocco dev'essere letto. La dimensione della registrazione letta dipende dalla RECORDSIZE precisata nell'istruzione OPEN. Essa può tuttavia essere inferiore alla dimensione minima del buffer (512 caratteri).

Esempio

```
100     GET # 3, RECORD 1% FOR 1% = 1% TO 100
        STEP 5%
110     GET # 4%
Ready
```

La scrittura del contenuto del campo di scambio con l'archivio si esegue mediante l'istruzione PUT:

PUT # espressione 1 [, RECORD espressione 2]

dove il significato delle espressioni 1 e 2 è identico a quello descritto in precedenza.
La dimensione del campo trasferito dipende, come nell'istruzione GET, dalla RECORDSIZE.
Nel caso in cui la lunghezza dell'articolo logico sia inferiore alla dimensione del buffer, un'istruzione GET (o PUT) può trasferire verso l'archivio numerose registrazioni logiche quando queste sono raggruppate. L'utilizzazione dell'opzione RECORD è identica a quella dell'istruzione GET.

Esempio

```
100     PUT # 3%
110     PUT # N%, RECORD 254%
Ready
```

4.4.2. Accesso al buffer degli archivi I/O per registrazione

La possibilità di trasferimento delle informazioni da e verso gli archivi non è sufficiente. È necessario in più avere accesso alle informazioni al fine di poterle consultare e modificare.
Queste funzioni sono assolte dalle istruzioni:

FIELD, LSET e RSET

La forma generale della prima istruzione è:

FIELD (espressione), espressione 1 AS vsc, espressione 2 AS vsc 2,....

ove: vsc sta per variabile stringa di caratteri
espressione definisce il n. di canale
espressione *n* definisce la lunghezza della variabile stringa di *n* caratteri.

Questa istruzione permette di creare dei legami logici tra i nomi delle variabili, stringhe di caratteri e la totalità o parti dell'area di trasferimento associata a un archivio. L'istruzione FIELD non ha alcuna azione fisica sulla registrazio-

ne; essa permette semplicemente di apporre una maschera a un'area di scambio.
Consideriamo una registrazione della lunghezza di 128 caratteri composta nel modo seguente:

```
100        OPEN «FICH.DAT» AS FILE 10%
110        FIELD # 10%, 30% AS N$, 20% AS P$, 78% AS A$
Ready
```

Nel seguito del programma, N$ corrisponderà ai primi 30 caratteri dell'articolo, P$ ai seguenti 20 caratteri ed A$ ai seguenti 78 caratteri P$.
Con l'istruzione FIELD dell'esempio non si definiscono che i primi 128 caratteri dell'area di trasferimento (ipotizzata qui uguale a 512 caratteri); i 384 caratteri che seguono non sono utilizzati.
Al fine di meglio utilizzare tutti i caratteri della registrazione fisica, è sufficiente raggruppare più registrazioni logiche nella stessa registrazione fisica.

```
100    OPEN «FICH.DAT» AS FILE 10%
110    FIELD # 10%, 30% AS N$ (0%), 20% AS P$ (0%), 78% AS
                   , A$ (0%)
                   , 30% AS N$ (1%), 20% AS P$ (1%), 78% AS
                   , A$ (1%)
                   , 30% AS N$ (2%), 20% AS P$ (2%), 78% AS
                   , A$ (2%)
                   , 30% AS N$ (3%), 20% AS P$ (3%), 78% AS
                   , A$ (3%)
Ready
```

L'istruzione FIELD della linea 190 è ugualmente scritta sotto la forma:

```
100        FIELD # 10%, I% * 128% AS Z$,
           30% AS N$ (I%),
           20% AS P$ (I%),
           78% AS A$ (I%)          FOR I% = 0% TO 3%
Ready
```

Una volta definite le variabili stringhe di caratteri sull'area di trasferimento, è possibile assegnare loro dei valori.
Questa operazione dev'essere svolta per mezzo delle istruzioni:

```
LSET <variabile stringa di caratteri> = <stringa>
RSET <variabile stringa di caratteri> = <stringa>
```

Queste due istruzioni collocano nell'area di trasferimento una costante o una variabile stringa di caratteri, cancellando in primo luogo il precedente conte-

nuto della variabile. Le istruzioni LSET e RSET non modificano la lunghezza della variabile precedentemente definita.
Così, se la nuova stringa è più lunga, essa verrà troncata; se al contrario essa è più corta, sarà completata con spazi bianchi.

— A destra con l'istruzione LSET
— A sinistra con l'istruzione RSET

In altre parole la nuova stringa di caratteri sarà collocata a sinistra con LSET e a destra con RSET.

4.4.3. Esempio di utilizzazione d'un archivio I/O per registrazione

Enunciato del problema

Occorre gestire un archivio del personale le cui registrazioni sono definite nel modo che segue:

— cognome 30 caratteri
— nome 20 caratteri
— indirizzo 78 caratteri

L'accesso all'archivio si realizza mediante un codice numerico il cui valore può variare da 1 a 1000.
Le operazioni da effettuare sull'archivio sono:

— creazione della registrazione completa
— modifica della registrazione campo per campo
— soppressione della registrazione

Il disegno della registrazione fisica e logica è dato nell'esempio precedente.

```
5       ON ERROR GOTO 32700
10      OPEN «PERSON.DAT» AS FILE 1%
20      FIELD # 1%, I% * 128% AS Z$, 30% AS N$ (I%), 20% AS P$
        (I%), 78% AS A$ (I%) FOR I% = 0% TO 3%
30      PRINT «(C)reazione, (M)odifica, (S)oppressione, (F)ine lavoro»
40      INPUT «vostra scelta»; C$
/       GOTO 32760 IF C$ = «F»
/       GOTO 40 IF C$ < > «C» AND C$ < > «M» AND C$ < > «S»
100     INPUT «MATRICOLA»; M%
/       GOTO 100 IF M% < 1% OR M% > 1000%
110     M1% = (M% — 1%) / 4% + 1%
/       M2% = M% — (M1% — 1%) * 4% — 1%
120     GET # 1, RECORD M1%
```

```
130        GOTO 200 IF C$ = «M» OR C$ = «S»
           !
           ! CREAZIONE
           !
140        IF N$ (M2%) < > SPACE$ (30%) THEN PRINT «MATRICO-
           LA ESISTENTE» / GOTO 30
150        INPUT «COGNOME»; N1$
/          GOTO 150 IF N1$ = «»
/          INPUT «NOME»; P1$
/          PRINT «INDIRIZZO»;
/          INPUT LINE A1$
/          A1$ = LEFT (A1$, LEN (A1$) — 2%)
160        LSET N$ (M2%) = N1$
/          LSET P$ (M2%) = P1$
/          LSET A$ (M2%) = A1$
170        PUT #1%, RECORD M1%
/          GOTO 30
           !
           !
200        IF N$ (M2%) = SPACE$ (30%) THEN PRINT «MATRICOLA
           INESISTENTE» / GOTO 30
210        PRINT
/          PRINT «1.COGNOME :»;N$(M2%)
/          PRINT «2.NOME :»;P$(M2%)
/          PRINT «3.INDIRIZZO :»;A$(M2%)
/          PRINT
/          GOTO 300 IF C$ = «S»
220        !
           ! MODIFICA
           !
230        INPUT «N. DI RUBRICA DA MODIFICARE»; I%
/          GOTO 280 IF I% < = 0%
/          GOTO 230 IF I% > 3%
240        ON I% GOTO 250, 260, 270
250        INPUT «NUOVO COGNOME»; N1$
/          GOTO 230 IF N1$ = «»
/          LSET N$ (M2%) = N1$
/          GOTO 230
260        INPUT «NUOVO NOME»; P1$
/          LSET P$ (M2%) = P1$
/          GOTO 230
270        PRINT «NUOVO INDIRIZZO»;
/          INPUT LINE A1$
/          A1$ = LEFT (A1$, LEN (A1$) — 2%)
/          LSET A$ (M2%) = A1$
/          GOTO 230
```

```
280       GOTO 320
300       !
          ! SOPPRESSIONE
          !
310       INPUT «SOPPRESSIONE (SI/NO)»; C$
/         GOTO 30 IF C$ < > «SI»
/         LSET N$ (M2%) = «»
/         LSET P$ (M2%) = «»
/         LSET A$ (M2%) = «»
320       PUT # 1, RECORD M1%
/         GOTO 30
32700     IF ERR = 50 OR ERR = 52 THEN RESUME
32710     IF ERR = 11 AND C$ = «C» THEN RESUME 150
          ELSE PRINT «MATRICOLA INESISTENTE» / RESUME 30
32760     CLOSE 1%
32767     END
Ready

RUNNH
(C)reazione, (M)odifica, (S)oppressione, (F)ine lavoro


Vostra scelta? C
MATRICOLA? 100
COGNOME? ROSSI
NOME? GIANNI
INDIRIZZO? VIA NINO BIXIO 27 MILANO
(C)reazione, (M)odifica, (S)oppressione, (F)ine lavoro
Vostra scelta? C
MATRICOLA? 100
MATRICOLA ESISTENTE
(C)reazione, (M)odifica, (S)oppressione, (F)ine lavoro
Vostra scelta? M
MATRICOLA? 100

1.COGNOME :ROSSI
2.NOME     :GIANNI
3.INDIRIZZO:VIA NINO BIXIO 27 MILANO

N. DI RUBRICA DA MODIFICARE? 2
NUOVO NOME? LUIGI
N  DI RUBRICA DA MODIFICARE?
(C)reazione, (M)odifica, (S)oppressione, (F)ine lavoro
Vostra scelta? F

Ready
```

Dopo l'apertura dell'archivio (linea 10) e la definizione della suddivisione dell'area di trasferimento (linea 20), si domanda all'utilizzatore di scegliere il tipo di operazione da effettuare e il numero di matricola.
Successivamente (linea 110), in funzione del numero di matricola, si calcola il numero della registrazione nella quale si colloca l'articolo in questione (M1%), oltre che la posizione relativa di questo articolo nell'ambito della registrazione.
Dopo aver letto l'archivio (linea 120), si controlla il tipo di operazione da effettuare; se si tratta di creazione, si controlla la disponibiltà della registrazione relativa alla matricola. Per questo è sufficiente controllare se l'area corrispondente al cognome è occupata (il cognome è l'unico campo obbligatorio).
Se il controllo è positivo si accettano le informazioni. L'immissione si esegue mediante le istruzioni INPUT per il cognome e il nome, e mediante l'istruzione INPUT LINE per l'indirizzo (possibilità di segni di punteggiatura che vengono considerati delimitatori di campo da INPUT). Poiché la variabile A1$ contiene il RETURN (ritorno carrello), questo viene soppresso alla linea successiva. Dopo la collocazione dei dati nell'area di trasferimento (linea 160), questi vengono scritti nell'archivio.
Le elaborazioni effettuate nelle parti MODIFICA e SOPPRESSIONE sono quasi identiche alla parte CREAZIONE.

4.5. GLI ARCHIVI IN MEMORIA VIRTUALE

Gli archivi in memoria virtuale permettono di elaborare i dati che occupano molto spazio in modo molto semplice ed efficace. In questi archivi i dati sono organizzati sotto forma di matrici. Prima di utilizzare un archivio di memoria virtuale è necessario aprirlo, come qualunque altro archivio, mediante l'istruzione OPEN. Il fatto che l'archivio si trovi in memoria virtuale è indicato al sistema dall'istruzione:

```
DIM (espressione), (lista)
```

espressione che definisce il numero del canale e la lista che ha lo stesso significato della lista d'una istruzione DIM classica.
Ad esempio, per utilizzare una matrice di numeri interi dimensionata 1000 × 100 si scriverà:

```
15 DIM # 10, Z$ (200%) = 64%, G$ (100%) = 2%, D$ (1000%)
Ready
```

È anche possibile utilizzare gli archivi in memoria virtuale contenenti stringhe di caratteri. A differenza della memoria centrale ove le stringhe di caratteri possono avere una lunghezza qualunque, gli archivi in memoria virtuale necessitano di una definizione di lunghezza fissa, che dev'essere una potenza di 2 inferiore o uguale a 512 (2,4,8,16,32,64,128,512). La definizione della lunghezza si ottiene mediante l'istruzione DIM, che si scrive:

DIM (espressione), stringa di caratteri (dimensione) = lunghezza

La lunghezza normale, scelta dal sistema quando nulla viene precisato, è di 16 caratteri.
Questa istruzione definisce la tabella Z$ di 201 (si può utilizzare anche l'elemento 0) stringhe di caratteri la cui lunghezza massima è uguale a 64 caratteri, la tabella G$ di 101 elementi di lunghezza massima di 2 caratteri e la tabella D$ contenente 1001 elementi di lunghezza di 16 caratteri.
Una volta definite con l'istruzione DIM le tabelle in memoria virtuale, si utilizzano come se fossero in memoria centrale.
Come qualunque archivio, anche quelli in memoria virtuale debbono essere chiusi alla fine dell'utilizzazione.

Esempio di programma

Il programma permette la creazione e la consultazione d'una tabella contenente le descrizioni di 10000 pezzi.

```
10      ON ERROR GOTO 32700
20      OPEN «PEZZI» AS FILE 5%
30      DIM # 5, N$ (10000%) = 32%
40      INPUT «N. PEZZO»; N%
/       GOTO 32700 IF N% < = 0%
/       GOTO 30 IF N% > 10000%
50      IF N$ (N%) < > «» THEN PRINT N$ (N%)
/       PRINT
/       GOTO 40
60      PRINT «PEZZO INESISTENTE»
/       INPUT «IMMETTERE LA DESCRIZIONE»; N$
        (N%)
/       PRINT
/       GOTO 40
32700   IF ERR = 50 OR ERR = 52 THEN RESUME
32760   CLOSE 5%
32767   END
Ready

RUNNH
N. PEZZO? 135
PEZZO INESISTENTE
IMMETTERE LA DESCRIZIONE? PEZZO N. 135
N. PEZZO? 135
PEZZO N. 135
N. PEZZO? 100
PEZZO 100
N. PEZZO?

Ready
```

CAPITOLO 5

PROBLEMI APPLICATIVI

5.1. UN METODO DI SELEZIONE

Enunciato

Organizzare N numeri in ordine crescente

Metodo

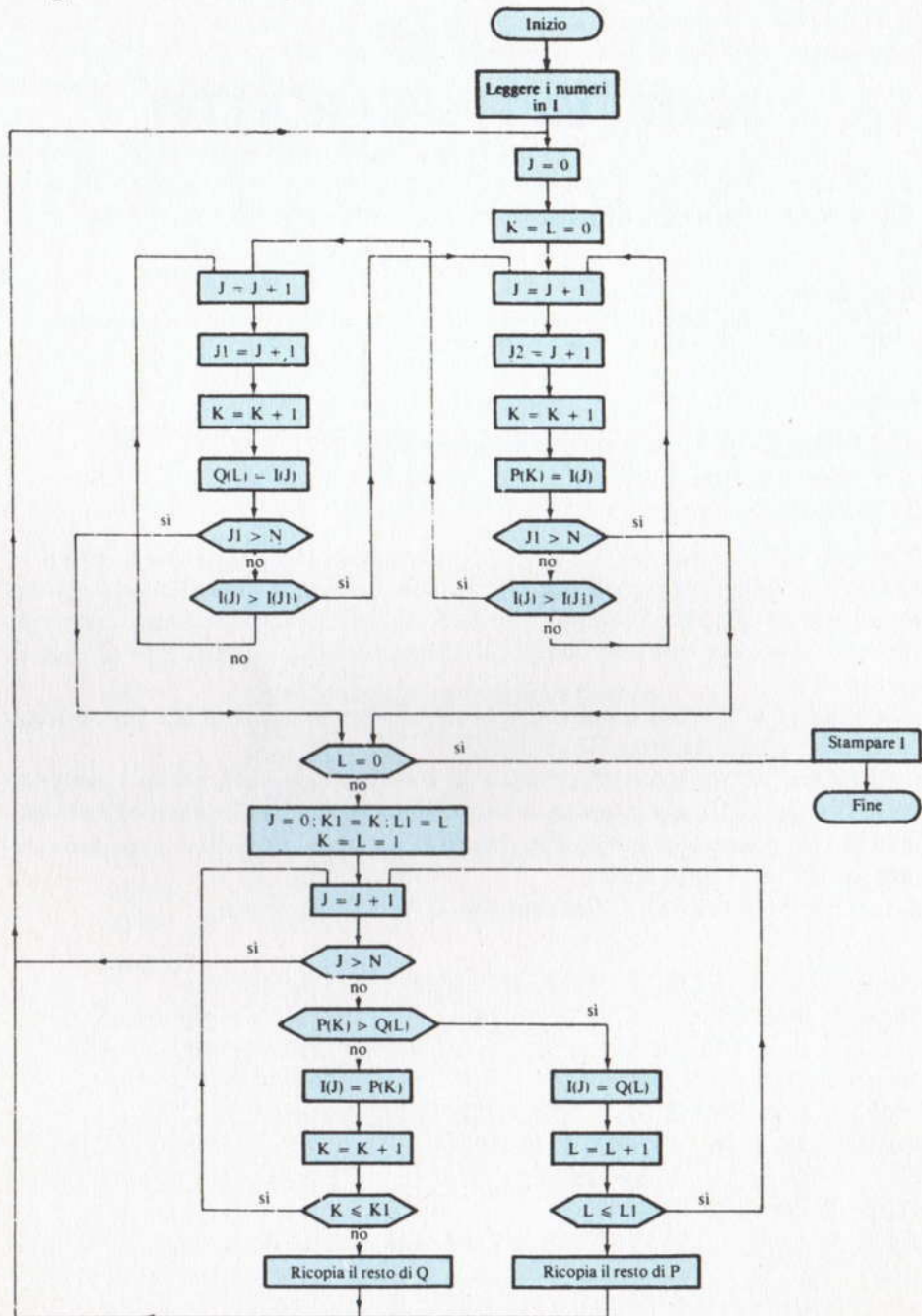
1 Si collocano gli N numeri in un vettore I(J)
2 Si prevedono due altri vettori P(K) e Q(L)
3 Si percorre il vettore I(J):

Fintanto che gli elementi sono in ordine crescente li si ricopiano in P(K). Quando la sequenza crescente s'interrompe, s'inizia la trascrizione degli elementi I(J) in Q(L) e si continua fintanto che questi sono in ordine crescente. Quando la sequenza s'interrompe, si ritorna a P(K); questo fino all'esaurimento di I.
4 Se alla fine non figura alcun elemento in Q, questo significa che I era ordinato. Si può allora stampare il vettore P.
5 Altrimenti si ricopiano gli elementi di P e Q successivamente in I partendo da P(1) e Q(1). A ciascun passo si confronta P(K) a Q(L) e si scrive il più piccolo in I(J). Quando si arriva alla fine di P o di Q si trasferisce il residuo contenuto nell'altro vettore in I.
6 Si ricomincia dall'inizio fintanto che Q non risulti vuoto.

Ad esempio: I = (45, 67, 1024, 473, 1028, 3, 18, 48, 19, 52)
Passo 3: P = 45, 67, 1024, 3, 18, 48
Q = 473, 1028, 19, 51
Passo 4: Q non è vuoto
Passo 5: I_1 = (45, 67, 473, 1024, 1028, 3, 18, 19, 48, 52)
Passo 3: P_1 = (45, 67, 473, 1024, 1028)
Q_1 = (3, 18, 19, 48, 52)
Passo 4: Q_1 non è vuoto
Passo 5: I_2 = (3, 18, 19, 45, 48, 52, 67, 475, 1024, 1028)
P_2 = (--------------------------------------)
Q_2 = 0

Nota: questo algoritmo interessa solo gli insiemi che sono già quasi ordinati. È il caso, ad esempio, di un archivio in cui siano state introdotte alcune nuove registrazioni in modo casuale.

Diagramma di flusso

Inizio
Leggere i numeri in I
J = 0
K = L = 0
J = J + 1
J1 = J + 1
K = K + 1
Q(L) = I(J)
J1 > N
I(J) > I(J1)
J = J + 1
J2 = J + 1
K = K + 1
P(K) = I(J)
J1 > N
I(J) > I(J1)
si
no
L = 0
Stampare I
Fine
J = 0: K1 = K: L1 = L
K = L = 1
J = J + 1
J > N
P(K) > Q(L)
I(J) = P(K)
K = K + 1
K ≤ K1
Ricopia il resto di Q
I(J) = Q(L)
L = L + 1
L ≤ L1
Ricopia il resto di P

Programma

```
 10 DIM I(40), P(40), Q(40)
 20 READ N
 30 FOR J = 1 TO N
 40 READ I(J)
 50 NEXT J
 60 J = 0
 61 K = 0
 62 L = 0
 70 J = J + 1 / K = K + 1 / J1 = J + 1
 80 P(K) = I(J)
 90 IF J1 — N 0 > THEN 270
100 IF I(J) — I(J1) > 0 THEN 180
150 GO TO 70
180 J = J + 1
190 L = L + 1
200 J1 = J + 1
210 Q(L) = I(J)
220 IF J1 — N > 0 THEN 270
230 IF I(J) — I(J1) > 0 THEN 100
240 GO TO 180
270 IF L = 0 THEN 560
290 J = 0 / K1 = K / L1 = L
300 K = 1 / L = 1
330 J = J + 1
340 IF J > N THEN 60
350 IF P(K) > Q(L) THEN 460
370 I(J) = P(K)
380 K = K + 1
390 IF K < = K1 THEN 330
400 FOR L2 = L TO L1
410 J1 = K1 + L2
420 I(J1) = Q(L2)
430 NEXT L2
440 GO TO 60
460 I(J) = 0(L)
470 L = L + 1
480 IF L < = L1 THEN 330
490 FOR K2 = K TO K1
500 J1 = L1 + K2
510 I(J1) = P(K2)
520 NEXT K2
530 GO TO 60
560 FOR K = 1 TO N
570 PRINT P(K)
```

```
580 NEXT K
590 STOP
600 DATA 10
610 DATA 45, 67, 1024, 473, 1028, 3, 18, 48, 19, 52
620 END
```

5.2. CATALOGAZIONE DI DATI STATISTICI

Enunciato

Si ipotizza di aver registrato una serie di dati numerici non ordinati. Si desidera evidenziare il numero dei dati immessi, la tabella delle frequenze assolute, relative e cumulate per determinate classi.

Dati

Per ciascuna serie, l'utilizzatore dovrà precisare:
- — il numero di classi
- — il limite superiore della 1ª classe
- — i differenti intervalli
- — la serie di dati

S'immetterà il numero 99999 per marcare la fine di una serie.

Programma

```
 C2  .BAS
 10 PRINT «ANALISI DEI DATI»
 20 PRINT PRINT
 30 S1 = 0
 40 S2 = 0
 50 D1 = 99999
 60 D2 = —99999
 70 FOR N = 1 TO 100
 80 INPUT D
 90 IF D = 99999 THEN 170
100 S1 = S1 + D↑
110 S2 = S2 + D2
120 IF D> = D1 THEN 140
130 D1 = D
140 IF D< = D2 THEN 160
150 D2 = D
160 NEXT N
170 N = N — 1
200 M = (D1 — D2)/2
205 IF N = 0 GOTO 240
210 M1 = S1/N
220 V = (N * S2 — S1↑ 2) / N
230 E = SQR(V)
```

```
240 PRINT «NUMERO DEGLI ELEMENTI DELLA SERIE»; N
250 PRINT «VALORE MASSIMO»; D2
260 PRINT «VALORE MINIMO»; D1
270 PRINT «MEDIANA»; M
280 PRINT «VALORE MEDIO»; M1
290 PRINT «SCARTO TIPO»; E
300 INPUT R
310 IF R = 0 THEN 10
400 END
```

5.3. TRACCIATO DI UN ISTOGRAMMA

Enunciato

Sia dato un gruppo di 1000 persone le cui altezze sono suddivise in sette classi:

160-165	50
165-170	300
170-175	430
175-180	110
180-185	30
185-190	0
190-195	80

Rappresentare l'istogramma di questa popolazione.

Metodo

A causa del modo di funzionamento della stampante o della visualizzazione sullo schermo è più facile programmare le classi verticalmente e le frequenze orizzontalmente.

Programma

```
 10 PRINT «ISTOGRAMMA»
 20 DIM N(7)
 25 M1 = 0
 30 K = 0
 40 FOR I = 1 TO 7
 50 READ N (I)
 55 IF M1 > N(I) THEN 60
 57 M1 = N(I)
 60 NEXT I
 70 FOR J = 160 TO 190 STEP 5
 80 K = K + 1
 90 PRINT «DA»; J; «A»; J + 5;
100 PRINT TAB (15); «I»;
120 FOR L = 1 TO N(K)/M1 * 50
130 PRINT «»;
140 NEXT L
150 PRINT N(K)
160 PRINT TAB(15) «I»
170 NEXT J
200 DATA 50, 300, 430, 110, 30, 0, 80
210 END
```

5.4. SOLUZIONE DI UN SISTEMA LINEARE

Enunciato

Determinare se un sistema d'equazioni lineari ammette soluzioni e calcolarle

Metodo

Sia dato il sistema:

A(1, 1)X(1) + A(1, 2)X(2) + ... + A(1, N)X(N) = A(1,N + 1)

....

A(I, 1)X(1) + A(I, 2)X(2) + ... + A(I, N)X(N) = A(I, N + 1)

....

A(N, 1)X(1) + ... + A(N, N)X(N) = A(N, N + 1)

1. X(1) è determinato dal valore delle altre incognite nella prima equazione. Lo si deve sostituire nelle equazioni seguenti con il suo valore, funzione delle altre incognite, estratto dalle altre equazioni: moltiplicando la prima equazione per A(I, 1) e la I-esima per A(1, 1) e sottraendo le due si ottiene una equazione nella quale manca X(1). Si elimina dunque X(1) dalle equazioni 2 ÷ N e si ottiene un sistema di N-1 equazioni lineari in N-1 incognite (X(2), X(3), ... X(N — 1)) e un'equazione che dà X(1) in funzione delle altre incognite.
Si ripete questo procedimento sul nuovo sistema. Si diminuisce dunque l'ordine del sistema da risolvere a ciascun passo fino all'ordine 1: si arriva infine a un'equazione di primo grado con incognita X(N).

2. Si ottiene un sistema «triangolare». Risolvendo l'ultima equazione si ottiene X(N). Successivamente si sostituisce X(N) con il valore trovato nelle equazioni da 1 ad N — 1 e si ottiene un sistema di ordine N — 1 in cui l'ultima equazione è una operazione di primo grado in N — 1.
Si ripete la procedura calcolando successivamente tutte le radici.

3. È impossibile risolvere il sistema con questo algoritmo quando nel secondo passo ci si ritrova con un'equazione della forma:

0 * X(1) = A(I, N + 1)

cioé se A(I, I) = 0
In questo caso si va a ricercare l'equazione seguente tale che il coefficiente di X(I) sia differente da zero; se non esiste, il sistema è impossibile; se invece esiste si permutano le due equazioni. Questo non è più possibile alla N-esima equazione per la quale, se il coefficiente è nullo, per X(N) non si può più permutare e il sistema risulta impossibile.

Nota: la permutazione tra il contenuto di due caselle si ottiene utilizzando una terza variabile qui definita B e che servirà ulteriormente più avanti. Il problema è simile a quello di scambiare il contenuto di due bicchieri colmi: ne serve un terzo.

Diagramma di flusso
Inizio
Lettura dei coefficienti
K = 0
K = K + 1
A(K, K) = 0
no
si
I = K
I = I + 1
I > N
A(I, K) = 0
si
no
Permutare
A(I, M) e A(K, M)
di M = K fino a N + 1
Per l'equazione I
Sostituire l'I-esimo coefficiente A(I, J) della equazione I con A(I, J) × A(K, K) — A(K, J) × f(/, K)
Questo per tutti i J da K + 1 a N + 1 e mettere A(I, K) a 0
Questo per tutti gli I da K + 1 ad N
K = N
no
si
A(N, N) = 0
si
no
Stampare impossibile
Fine
Stampare possibile
K = N + 1
K = K - 1
Calcolare e stampare X(K) = A(K, N + 1) / A(K, K)
Sostituire A(I, N + 1) con A(I, N + 1) — A(I, K) × X(K)
questo per tutte le I da 1 a K — 1
no
K = 1
si
Stampare X1 = A(1, N + 1) / A(1, 1)
Fine

Programma

```
 10 READ N
 20 DIM A(4, 5)
 30 FOR I = 1 TO 4 : FOR J = 1 TO 5
 40 READ A(I, J) : NEXT J : NEXT I
 50 FOR K = 1 TO N — 1
 60 IF A(K, K) < > 0 THEN 170
 70 FOR I = K + 1 TO N
 80 IF A(I, K) < > 0 THEN 110
 90 NEXT I
100 GOTO 380
110    FOR M = K TO N + 1
120    LET B = A(I, M)
130    LET A(I, M) = A(K, M)
140    LET A(K, M) = B
150    NEXT M
160
170    FOR I = K + 1 TO N
180          FOR J = K + 1 TO N + 1
190          LET A(I, J) = A(I, J) * A(K, K) — A(K, J) * A(I, K)
200          NEXT J
210    LET A(I, K) = 0
220    NEXT I
230 NEXT K
240 IF A(N, N) = 0 THEN 380
250
260 PRINT «SISTEMA POSSIBILE LE SOLUZIONI
    SONO:»
270 FOR K = N TO 2 STEP — 1
280 LET B = A(K, N + 1)/A(K, K)
290 PRINT «X»; K; « = »; B
300    FOR I = 1 TO K — 1
310    LET A(I, N + 1) = A(I, N + 1) — A(I, K) * B
320    NEXT I
330 NEXT K
340 LET X1 = A(1, N + 1)/A(1, 1)
350 PRINT «X1 = »; X1
360 STOP
370
380 PRINT «SISTEMA IMPOSSIBILE»
390 STOP
400 DATA 4
410 DATA 1, 2, —1. —2, 0
```

```
420 DATA 1, 1, —1, —1, 4
430 DATA 1, 3, —3, —1, 2
440 DATA 3, 2, 1, 1, 5

RUN

SISTEMA POSSIBILE LE SOLUZIONI SONO:
X 4 =        .142857
X 3 =      —2.85714
X 2 =      —3.85714
X 1 =       5.14286

USED        3.33 SEC.
```

5.5. CALCOLO DELL'INTERESSE COMPOSTO

Enunciato

Ammettiamo di aver depositato in banca la somma S al tasso I. Si chiede di calcolare il valore del capitale al termine di N anni.

Analisi del problema

Si calcola S1 = S * (1 + I)N
Fare riferimento ai paragrafi 1 e 2.

5.6. RECUPERO DI SPAZIO IN MEMORIA. ITERAZIONI

Enunciato

Il sistema lineare sia già triangolarizzato (si veda il paragrafo 5.4.):

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= a_{1,\ n+1} \\ a_{22}x_2 + \cdots + a_{2n}x_n &= a_{2,\ n+1} \\ &\text{----------------} \\ a_{nn}x_n &= a_{n,n+1} \end{aligned}$$

La soluzione può essere ricercata in modo identico al problema 5.4. In questo caso si risparmia il più possibile la memoria. Una prenotazione in memoria per A(N, N + 1) farebbe perdere molto spazio: la metà dei valori è nulla e quindi inutile.
Trovare un metodo per collocare i coefficienti non nulli in un vettore a una dimensione, e risolvere il sistema con questa disposizione. Fare riferimento ai paragrafi 2.3, 2.4, 2.5 e 2.6.

5.7. CALCOLO DELL'IMPOSTA

Enunciato

Dato un salario annuale S da dichiarare e data una composizione familiare (immessi con una istruzione INPUT, ad esempio) calcolare l'imposta sul reddito nel caso generale più semplice.

5.8. SOLUZIONE DI UN'EQUAZIONE CON IL METODO DI NEWTON

Enunciato

Si debba risolvere l'equazione: $F(x) = 0$; il metodo di Newton si basa sul fatto che se x_i è un'approssimazione della radice x_0 della equazione,

$$x_{i+1} = x_i - \frac{F(x_i)}{F'(x_i)}$$

è un'approssimazione molto migliore di x_0 [$F'(x)$ è la derivata di $F(x)$].

Analisi del problema

Consideriamo l'esempio:

$$F(x) = \text{sen } x - 0{,}3$$

Si cerchino gli angoli x tali che la loro funzione seno sia uguale a 0,3. In prima approssimazione l'angolo è compreso tra 0 e 1 radiante. Assumiamo:

$$x_1 = 0{,}5$$

Nel nostro caso $F'(x) = \cos x$.
La formula di Newton dà:

$$x_{i+1} = x_i - \frac{\text{sen } x_i - 0{,}3}{\cos x_i}$$

Si realizzerà nel programma un'iterazione nella quale si sostituirà

$$X \text{ con } \left(X - \frac{\text{sen } X - 0{,}3}{\cos X} \right)$$

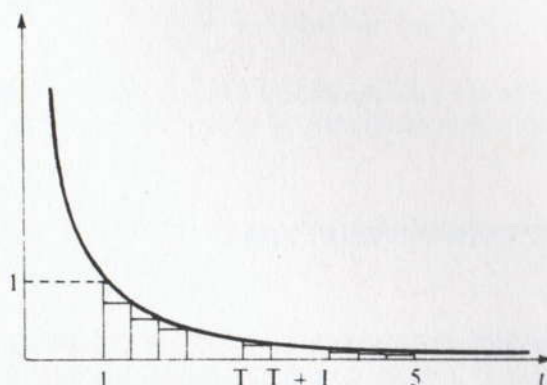
e ci si fermerà quando (sen X — 0,3) risulterà minore di una data costante. Si vedano i paragrafi 2.3 e 2.7.2.

5.9. CALCOLO DI UN INTEGRALE

Enunciato

Calcolare

$$\int_1 \frac{1}{t}\,dt.$$

Nota: si potrà ad esempio calcolare il valore dell'integrale per x = 5. Si fa un campionamento dell'asse delle t in intervalli parametrizzati I.

e si calcolerà la somma finita

$$\sum_{T=1}^{5} \frac{1}{T} * I$$

ove T è incrementato di I in ciascun intervallo (T = T + I). Questa somma finita rappresenta *la somma delle superfici dei rettangoli elementari della figura*: essa approssima

$$\int_1^5 \frac{1}{T}\,dT$$

che è la superficie compresa tra la curva e l'asse delle T tra i punti T = 1 e T = 5.
Si sa che

$$\int_1^x \frac{1}{t}\,dt = \log x$$

Si confronterà dunque, per un dato valore di I,

$$\sum_{T=1} \frac{1}{T} * I$$

a log 5, quest'ultimo essendo calcolato grazie alla funzione log(x) che si trova in biblioteca. Si potrà studiare la differenza tra

$$\sum_{T=1} \frac{1}{T} * I$$

e log 5, in funzione di I e si determinerà il valore di I che porta la differenza al livello di precisione del calcolatore. Si vedano i paragrafi 2.3 e 2.7.2.

5.10. PIANO RISPARMIO EDILIZIO

Enunciato

Data una rata mensile M e un versamento iniziale I, calcolare l'ammontare del prestito accordato al termine dei 4 anni, il rimborso mensile per N anni e la somma totale di cui si dispone al termine dei 4 anni per comprare un appartamento con il sistema del risparmio edilizio utilizzato dalle banche francesi dal 1970.

Note

1. L'interesse acquisito durante i 4 anni è dell'8% e rappresenta la somma I_1. L'interesse sul rimborso del prestito accordato è del 5,5% e rappresenta la somma I_2.
La formula che dà l'ammontare del prestito al termine dei 4 anni è

$$I_2 = 2,5 \times I_1$$

2. La somma totale di cui si dispone al termine dei 4 anni è data da:

a) il versato durante i 4 anni:

$$(4 \times 12 \times M) + I;$$

b) l'interesse dell'8% acquisito durante questi 4 anni;

$$I_1 = (0,08 \times 4 \times I) + \sum_{k=0}^{47} M(48 - k) \frac{0,08}{12}$$

Si utilizzerà per il calcolo di questa somma l'uguaglianza

$$\sum_{i=1}^{n} (i) = \frac{n(n-1)}{2};$$

c) l'ammontare del prestito accordato P (tale che $I_2 = 2,5 \times I_1$).

3. Il calcolo di I_2, se si rimborsa P in rate M_1 mensili durante N anni, è tale che:

$$M_1 = P \frac{i}{12} \frac{(1+i)^N}{(1+i)^N - 1} \text{ (qui i = 5,5/100)}$$

Dunque $I_2 = (48 \times M_1) - P$

5.11. RICERCA DI UNA SEQUENZA DI CARATTERI PREDEFINITI

Enunciato

Sia dato un archivio binario; il problema consiste nel trovare tutti i gruppi di 5 zeri successivi (il numero del primo...). Attenzione: 10 zeri successivi non sono due gruppi di 5 zeri; è necessario che prima e dopo i 5 zeri si trovi almeno un 1.

Note

1. È necessario trovare il metodo più rapido possibile che conduca al programma più corto possibile. Si faccia attenzione alla fine dell'archivio.

2. È necessario aver studiato l'appendice 2 per svolgere questo problema.

5.12. IL GIOCO DELLA DAMA: PROGRAMMAZIONE NON NUMERICA

Enunciato

Sia data una certa posizione nel gioco della dama: ci si colloca a metà gioco in quanto l'inizio costituisce un caso particolare.
La scacchiera può essere rappresentata da una matrice A(10, 10). Domandare all'utente due numeri I, J che rappresentano una casella della scacchiera A(I, J). E, secondo il caso, stampare:

1. *nessuna pedina;*
2. *pedina* Amica *(o* Nemica*); può avanzare in (I', J'), (I'', J'');*
3. *pedina* Amica *(o* Nemica*); prende le pedine situate in*

(I, J), (I', J'), ...

o in

$(I_1, J_1), (I'_1, J'_1)$, ...

Si suppone che non vi sia ancora alcun damone sulla scacchiera.

Analisi del problema

Si può decidere che una casella vuota sia rappresentata da A(I, J) = 0, una pedina amica per A(I', J') = 1 e nemica per A(I'', J'') = −1.
Il primo caso non offre alcuna difficoltà. Si noti che si dovrà controllare che i numeri I e J immessi siano compresi tra 1 e 10.
Il secondo caso offre degli aspetti interessanti: si faccia attenzione ai bordi della scacchiera...
Il terzo caso è difficile e richiede numerose iterazioni inscatolate l'una dentro l'altra. È però molto interessante e questo metodo, detto «ad albero», è molto utilizzato nella programmazione non numerica. Si potrà ipotizzare che non si possano mangiare più di 5 pedine, e che una data pedina non possa avere più di 4 modi differenti di presa... Se questi limiti vengono superati: fare stampare «voi disperdete troppo le vostre pedine, il vostro gioco non ha alcun interesse...».
Si veda il paragrafo 2.6.

5.13. METODO PER GENERARE DEI NUMERI PRIMI DELLA SERIE DI FIBONACCI

Enunciato

Ottenere una sequenza di numeri primi a partire dalla sequenza definita da:

$$F_i = F_{i-1} + F_{i-2}$$

dove F_i è l'i-esimo numero della sequenza e dove $F_1 = F_2 = 1$ (è una serie di Fibonacci).

Metodo

Un numero è detto primo se non è divisibile che da se stesso e dalla unità. Ad esempio 4 non è primo poiché i suoi divisori sono 1, 2 e 4. Invece 5 è primo poiché non ha altri divisori oltre l'1 e il 5. Per sapere se un numero intero è primo è sufficiente analizzare se esiste un numero Q tale che

$$Q = F/J \text{ e se } Q_1 = \text{INT}(Q) \text{ (parte intera di Q)}$$

allora è necessario che Q = Q1.

Programma Basic

```
 10 PRINT «N =» : INPUT N
 20 PRINT
 30 LET F1 = 1 : LET F2 = 1
 40 PRINT «SERIE DI N NUMERI PRIMI»
 45 PRINT «DI FIBONACCI»
 50 PRINT 1
 60 PRINT 2
 70 FOR I = 3 TO N
 80 F = F1 + F2
 90 FOR J = 2 TO F — 1
100 Q = F/J
110 Q1 = INT(Q)
120 IF Q = Q1 THEN 160
130 NEXT J
140 PRINT F
150 GOTO 170
160 F2 = F1
170 F1 = F
180 NEXT I
190 END
```

Risultati

```
N = ? 30
SERIE DI N NUMERI PRIMI DELLA SERIE DI FIBONACCI
     1
     2
     3
     5
    13
    89
   233
  1597
 28657
514229
```

5.14. FUNZIONE FATTORIALE. PERMUTAZIONI E COMBINAZIONI

Enunciato

Determinare il valore di N! poi il numero di permutazioni di N oggetti presi a gruppi di P ed infine il numero di combinazioni di N oggetti presi a gruppi di P.

Metodo

Si sa che $N! = 1 \times 2 \times \cdots N$
Il numero di permutazioni A_n^p di n oggetti a gruppi di p è dato da:

$$A_n^p = n(n-1)(n-2)\cdots(n-p+1)$$

cioé:

$$A_n^p = \frac{n!}{(n-p)!}$$

Ed infine il numero di combinazioni di n oggetti a gruppi di p è fornito dalla formula:

$$C_n^p = \frac{n!}{p!(n-p)!} = \frac{A_n^p}{p!}$$

Programma 1

```
 10 PRINT «DARE I VALORI DI N E DI P»
 20 INPUT N, P
 30 A = 1
 40 FOR T = 1 TO N
 50 A = A * T
 60 NEXT T
 70 PRINT «FATTORIALE», N, « = », A
 80 K = N — P
 90 B = 1
100 FOR I = 1 TO K
110 B = B * I
120 NEXT I
130 C = A/B
140 PRINT «PERMUTAZIONI DI», N, «OGGETTI», P,
    «A», P, « = », C
150 D = 1
160 FOR J = 1 TO P
170 D = D * J
180 NEXT J
190 E = C/D
200 PRINT «COMBINAZIONI DI», N, «OGGETTI», P, «A», P,
    « = », E
210 END
```

Programma 2

Sarebbe stato vantaggioso utilizzare il concetto di sottoprogramma. Cioé:

Sottoprogramma

```
      FAC
   10 A = 1
   20 FOR I = 1 TO M
   30 A = A * I
   40 NEXT I
   50 RETURN
```

Programma principale

```
   10 PRINT «DARE I VALORI DI N E DI P»
   20 INPUT N, P
   30 M = N
   40 CALL FAC
   50 PRINT «FATTORIALE», N, «=», A
   60 B = A : M = N — P
   70 CALL FAC
   80 C = B/A
   90 PRINT «PERMUTAZIONI DI», N, «OGGETTI», P, «A», P,
      «=», C
  100 M = P
  110 CALL FAC
  120 PRINT «COMBINAZIONI DI», N, «OGGETTI», P, «A», P,
      «=», C/A
  130 END
```

5.15. SIMULAZIONE DEL GIOCO DEL LOTTO

Enunciato

Ci si propone di eseguire un'estrazione casuale di 7 numeri compresi tra i numeri interi da 1 a 49.

Metodo

Si utilizza la funzione biblioteca RND che fornisce un numero casuale compreso tra 0 ed 1. Ma questa funzione è pseudo-aleatoria, e sarà quindi necessario assicurarsi che lo stesso numero non si ritrovi più volte in una stessa estrazione.

Programma

```
 10 DIM A(7)
 20 PRINT «VOLETE ESEGUIRE UNA ESTRAZIONE?»
 30 INPUT C$
 40 IF C$ = «NO» THEN 140
 50 PRINT «ESTRAZIONE DELLA SETTIMANA»
 60 FOR J = 1 TO 7
 70 A(J) = INT(49 * RND(25) + 1)
 80 NEXT J
 90 FOR I = 1 TO 6
 95 IF A(I) < > A(I + 1) THEN 110
100 A(I + 1) = INT(49 * RND(25) + 1)
105 GOTO 95
110 NEXT I
120 FOR I = 1 TO 6 : PRINT A(I) : NEXT I
130 PRINT «N. COMPLEMENTARE :», A(7)
140 PRINT «ALLA PROSSIMA VOLTA»
150 END
```

APPENDICE 1

Riepilogo delle istruzioni Basic

Istruzioni	Forma delle istruzioni corrispondenti	Riferimenti
DATA	10 DATA, 30, —20, 4.3	21, 22
DEF	10 DEF LOT(A) = (49 * RND(1) + 1)	48
DIM	10 DIM A(5, 10), B(30), C$(15)	37, 38
END	100 END	26
FOR... TO...	10 FOR I = 1 TO 30 STEP 5	
.	.	
.	.	37-45
.	.	
NEXT	100 NEXT I	
GOTO	10 GOTO 150	25, 27
GOSUB	10 GOSUB 300	49-52
IF... THEN	10 IF A > = B THEN 100	31-35
INPUT	10 INPUT A, C$	27, 28
LET	10 LET X = (— B + SQRD)/2 * A	23
PRINT	10 PRINT «X =»; X, Y(J)	25, 27, 29, 34
READ	10 READ K, A(I), B(I, J), N$, M$(I)	
REM	10 REM RADICI DELL'EQUAZIONE	
RETURN	10 RETURN	
STOP	1000 STOP	

Estensioni del Basic

CHAIN	17000 CHAIN 31000	59
CHANGE	10 CHANGE A$ TO A	
FNEND	40 FNEND	77
INPUT LINE #	10 INPUT LINE # 1, A$	84
IF-THEN-ELSE	10 IF A + B = 0 THEN 120 ELSE A + B = K	67-69
ON ERROR-GOTO	10 ON ERROR GOTO 32700	74-75
ON-GOTO	10 ON X GOTO 100, 110, 130	67
ON-GOSUB	10 ON K GOSUB 100, 150	68
RANDOMIZE	10 RANDOMIZE	
RESTORE	10 RESTORE	
FOR-STEP-WHILE	10 FOR X = 1 STEP2 WHILE Z < 30	70
FOR-STEP-UNTIL	10 FOR X = 1 STEP3 UNTIL Z > = 30	71

Gestione degli archivi

Istruzioni	Forma delle istruzioni corrispondenti	Riferimento
FIELD #	100 FIELD # 10%, 30% AS N$	88
GET #	100 GET # 3, RECORD I % FOR I% = 1% TO 100% STEP 5%	86
CLOSE	100 CLOSE 1ß, 7%	83
OPEN	100 OPEN «TOTO» AS FILE 2%	80
PUT #	100 PUT # N% RECORD 254%	87
DIM #	15 DIM # 10, Z$ (200%), ...	94

APPENDICE 2

Le funzioni biblioteca

FUNZIONE	ESEMPIO
ABS	10 LET X = ABS(Y)
ATN	10 LET Y = ATN(X)
ASC	10 LET N = ASC(M)
CHR$	10 LET N$ = CHR$(N)
COS	10 LET Y = COS(X)
COT	10 LET Y = COT(X)
EXP	10 LET Y = EXP(X)
INT	10 LET Y = INT(X)
LOG	10 LET Y = LOG(X)
RND	10 LET Y = RND(X) con X = intero qualunque
SIN	10 LET Y = SIN(X)
SQR	10 LET Y = SQR(X)
TAB	10 PRINT TAB(N);X
TAN	10 LET Y = TAN(X)

Bibliografia

J. ARSAC, *Les systèmes de conduite des ordinateurs,* Dunod.

J.P. BOUHOT *et al., Un fil d'Ariane,* Tomes I et II, Editions de l'Informatique.

B. DELVALLÉE, *Comment fonctionne un ordinateur; les systèmes d'exploitation,* Dunod.

L. NOLIN, *Formalisation des notions de machine et de programme,* Dunod.

J.C. SIMON, *Introduction à la structure et au fonctionnement des ordinateurs,* Masson et Cie.

R. ZAKS, *Introduction aux microordinateurs individuels et professionnels,* SYBEX (1978).

C. PARIOT, *Introduzione ai microprocessori e ai microelaboratori,* Tecniche Nuove.

L.A. LEVENTHAL, I. STAFFORD, *I piccoli calcolatori (personal computer),* Tecniche Nuove.

D. MARTIN, *Banca dati - applicazioni sui piccoli e grandi calcolatori,* Tecniche Nuove.

J.P. LAURENT, *Analisi e programmazione strutturata,* Tecniche Nuove.

J. RIVIÈRE, *Programmare in Assembler,* Tecniche Nuove.

C. MACCHI, J.F. GUILBERT, *Telematica, trasporto e trattamento dell'informazione,* Tecniche Nuove.

G. REALINI, *Disegnare col computer,* Tecniche Nuove.

B. MAYOH, *Programmare con il linguaggio ADA,* Tecniche Nuove.

L. POOLE, *IBM Personal Computer,* Tecniche Nuove.

Rivista CHIP, mensile di micro e personal computer, Tecniche Nuove.

Speciale 1

CHIP

Mensile di micro e personal computer

BASIC

corso di programmazione per microcalcolatori

BASIC

tecniche nuove

L. 8000